Trabajo Fin de Grado
Ingeniería de Electrónica, Robótica y Mecatrónica

Diseño y realización de un Autómata Programable basado en el microcontrolador Hercules RM46

Autor: Sergio Romero Ordieres

Tutor: Manuel Ángel Perales Esteve

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Ingeniería de Electrónica, Robótica y Mecatrónica

Diseño y realización de un Autómata Programable basado en el microcontrolador Hercules RM46

Autor:
Sergio Romero Ordieres

Tutor:
Manuel Ángel Perales Esteve

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Diseño y realización de un Autómata Programable basado en el microcontrolador Hercules RM46

Autor: Sergio Romero Ordieres
Tutor: Manuel Ángel Perales Esteve

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

PLC son en inglés las siglas de Programmable Logic Controller. Conocido en español como Controlador Lógico Programable o Autómata Programable. Un PLC se define como una computadora usada normalmente en la industria y el ámbito de la ingeniería para controlar y automatizar diferentes procesos. Un PLC se encarga básicamente de controlar las maquinas que estén implicadas en el proceso gracias a que primero recoge datos, pudiendo venir de sensores, botones o diversas señales de entrada, y actúa en congruencia para controlar los diferentes actuadores que pueda tener la máquina.

Por su importancia a la hora de la automatización, es muy común encontrarlos por ejemplo en fábricas donde se llevan a cabo procesos de montaje en línea. Aparte, están preparado para soportar condiciones adversas que pueden producirse en una fábrica como puede ser subidas altas de temperatura, humedad, resistencia a golpes o ruido eléctrico que pueda interferir en las señales que trata. También cabe destacar la longevidad de sus procesadores y el gran número de entradas y salidas que pueden gestionar.
Los PLCs tienen la obligación de trabajar en tiempo real. Deben responder a las entradas que han procesado en un tiempo específico para que el funcionamiento del sistema sea el correcto. Lo que convierte a un sistema digital en un sistema en tiempo real (STR) es la capacidad para interactuar con lo que le rodea, con sus entradas, salidas y tiempos críticos asegurando un buen funcionamiento del sistema. Esto no quiere decir que los sistemas en tiempo real sean rápidos o lentos, si no que se rigen por un tiempo límite en el que tienen que dar una respuesta.

Hoy en día los PLCs están preparados para entender muchos tipos diversos de programación. Entre ellos podemos encontrar desde el lenguaje Ladder, un lenguaje gráfico sencillo que permite a cualquiera entenderlo sin tener mucha formación sobre programación, pasando por diagramas de contactos hasta llegar a lenguajes de programación más avanzados como puede ser C. A fin de cuenta, todos ellos tienen el mismo propósito que es controlar el PLC. La función básica cuando se creó era reemplazar a sistemas de relés. También por esa razón uno de los lenguajes que se usan es el anteriormente comentado Ladder, el cuál es similar a un esquemático con la lógica de un relé. Con el tiempo, las funciones del PLC han ido cambiando y am-

pliando. Tanto es notable el cambio que los campos en los que se usan varían entre maniobras de instalaciones, plantas petroquímicas y químicas, metalurgia, producción de energía, etc.

Un PLC tiene diferentes partes. Estas pueden estar separadas por módulos o estar integradas juntas. Las principales partes son la fuente de alimentación, la unidad de procesamiento central (CPU), y los módulos de entradas y salidas. Entre los distintos tipos de PLC que existen, el que tiene relación con este proyecto ya que se basa en la idea de su distribución es el PLC Modular. Como su propio nombre indica, tiene sus diferentes partes divididas por módulos, dando la posibilidad de añadir más módulos si uno lo desea.
En este proyecto, se ha intentado seguir la misma distribución, manteniendo por separado la alimentación, la CPU que en nuestro caso se trata del microcontrolador Hercules RM46x, las entradas y las salidas. También se ha hecho una diferencia entre entradas digitales y analógicas. Para ello se ha separado cada 'módulo' en una placa electrónica diferente. En los próximos apartados explicaremos con más detalle los componentes de cada una de las partes.

Por último, mencionar que en este trabajo también se incluirá la pantalla VM800 para darle una parte más visual a los procesos. Se incluirá como componente de la placa principal y se explicará en dicho apartado.

Capítulo 2

Estado del arte

En este capítulo se explicará con mayor profundidad el concepto de Controlador Lógico Programable (PLC), se detallará como suelen ser y se expondrán ejemplos de algunos no tan comunes.

2.1. Historia

La idea de crear un aparato que pudiera controlar un sistema y que fuera programable vino por la necesidad que surgió de reemplazar a los sistemas basados en relés. Era lo más utilizado a finales de los años 60. Cuando se quería implementar algún sistema con una lógica combinacional se usaban circuitos basados en relés. Estos sistemas eran costosos ya que se requerían muchos relés para conseguir el resultado deseado, eran muy grandes por todos los cables que se necesitaban para unir todos los relés con sus correspondientes interruptores y más componentes y solían tener una vida no muy larga. Su tiempo de vida no era ilimitado dado que son dispositivos mecánicos. Por la necesidad de resolver todos estos inconvenientes, apareció la idea de un Controlador Lógico Programable.

Figura 2.1: Primer PLC de la historia

Así en 1968 apareció MODICON 084. Fue el primer PLC de la historia. Lo creó la empresa Bedford Associates y se produjo con intenciones comerciales. Un aparato resistente que no necesitara mucho mantenimiento, con una vida mucho más larga que los relés y fácil de programar. Esto último era una característica muy importante ya que los hacía útiles para más de un sistema. Podían ser programados para lo que se quisiera mientras que con los relés el circuito quedaba inutilizado teniendo que crear otro totalmente distinto para otro sistema.
Los PLCs avanzaron de tal forma que en el año 1973 eran capaces de comunicarse entre ellos. Esto los hacía aún más útiles al poder controlar alguna máquina o sistema sin estar si quiera cerca. Poco a poco fueron evolucionando aún más con respecto a las comunicaciones y a la manera de programarlos. Sin embargo, con el tiempo han perdido una batalla contra los microprocesadores. En los inicios de los PLCs estos eran más veloces pero con los años la cosa ha dado la vuelta hasta el punto en el que algunos PLCs se han visto reemplazados por ordenadores personales.

Aquí entra en juego la idea de este proyecto. Queremos hacer un PLC con un microcontrolador porque hoy en día las características de los microcontroladores son mucho mejores. En definitiva la idea es hacer un PLC mejorado.

2.2. Tipos de PLC

Cuando alguien esta pensando en comprar un PLC y se ha decidido finalmente por una marca específica, no es tan fácil escoger un modelo concreto. La gama de PLCs que existe es amplia y por tanto escoger el indicado no es tan sencillo. Existen PLCs de distintos precios y tamaños, esto quiere decir que algunos tendrán mejores características que otros. Puede ser que queramos un PLC para un sistema pequeño y fácil de controlar o para una fábrica donde se realizan cadenas de montaje. Como comprador tienes que ver que necesitas para comprar el producto lo más barato posible pero que cumpla tus expectativas y sacie tus necesidades. En el ejemplo antes expuesto, no sale rentable comprar un PLC con muchas entradas y salidas y con una gran cantidad de almacenamiento si tu sistema es pequeño y la mayor parte del PLC se va a desperdiciar. A parte de las características que normalmente se le piden a un PLC (velocidad, memoria, contadores, tension nominal), existen otras muy importantes que deben ser consideradas. Algunos de los factores de los no mencionados con anterioridad que pueden ser útiles son:

- Lenguajes permitidos
- Tipos de software permitido
- Actualizaciones de firmware
- Estabilidad de las versiones
- Cantidad y tipos de módulos de comunicaciones y señal

Normalmente las empresas que crean PLCs hacen una distinción entre gama básica y gama alta. La gama más cara como es obvio ofrece características. Sacan

modelos nuevos con sistemas operativos que sean más potentes, con una capacidad de almacenamiento mayor y con mayor velocidad de procesador. Los modelos más nuevo suele tener actualizaciones de firmware para mejorar el equipo y una gran estabilidad de las versiones, aunque en ocasiones puede ser que haya alguna incompatibilidad. De ser así quiere decir que ese modelo de PLC está anticuado para esas nuevas actualizaciones que la empresa esta desarrollando y quedará en una gama de más bajo nivel.

También cabe destacar la variedad de modelos que existe. Aquí por ejemplo tenemos 2 modelos diferentes. A la izquierda tenemos un PLC compacto y a la derecha un PLC modular.

Figura 2.2: PLC compacto

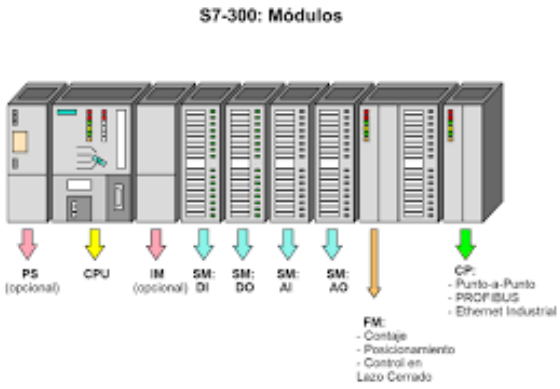


Figura 2.3: PLC modular

En principio si ambos modelos son del mismo fabricante y tienen las mismas características no habría diferencia ninguna entre los dos tipos de PLC. Sin embargo, hay una diferencia clara entre ambos. El PLC compacto tiene todas sus partes juntas. Esto quiere decir que no puede mejorar. No se le pueden añadir nuevas entradas ni salidas ni por ejemplo se le puede añadir más memoria. Con los PLCs modulares esto no pasa. Tienen la capacidad de encadenar más módulos lo que les da la capacidad de mejorar si la situación lo requiere. Si usamos PLCs compactos y de repente nos damos cuenta de que nos hemos quedado cortos con los pines de entrada obligatoriamente tendremos que comprar uno con más entradas y salidas. Con nuestro PLC modular, lo único que tendríamos que hacer es poner otro módulo más al conjunto.

Por este motivo en este trabajo se ha querido simular un PLC modular en el que podemos añadirle las entradas y salidas que queramos.

Capítulo 3

Arquitectura general de diseño

Como ya se ha mencionado anteriormente, la arquitectura general para el diseño de este proyecto esta basada en un PLC modular. Para seguir la misma idea, hemos construido cuatro placas electrónicas:

o de alimentación (es la que unimos al microcontrolador Hercules), placa con entradas analógicas, placa con entradas digitales y placa con salidas. Estas placas es el símil a los módulos de un PLC, consiguiendo así que cada parte pueda unirse o no al conjunto final.

En nuestro caso, no podíamos hacer algo tan genérico, con la capacidad para poder aceptar otros módulos. Este inconveniente nace por la idea de conexión entre la placa principal y el microcontrolador. Según cuantas entradas y salidas tuviera, esta placa tendría que estar conectada a más o menos pines del microcontrolador. Esto supone un problema que no se puede resolver por lo que se optó por poner un número estándar de ocho entradas digitales, ocho entradas analógicas y ocho salidas. Podría parecer lógico pensar que sería mejor poner el doble de salidas para que se iguale con el número de entradas pero existe un problema. El micro que estamos usando no posee salidas analógicas. Por esta razón, usamos solamente ocho salidas las cuáles responderán tanto a entradas digitales como a analógicas. En el caso de tener una entrada analógicas se usará un pwm para simular una salida analógica. Dicho esto es lógico comentar que pines del micro se usarán. Adelantar que aunque parezca raro los pines elegidos, se ha hecho así sobre todo por la distribución de pines. Aprovechando que hay tantos pines a nuestra disposición me ha parecido conveniente tenerlo en cuenta para que la placa principal quedara con una distribución más sencilla.

Los pines usados para las salidas son pines específicos para usar pwm. Con este microcontrolador también se pueden usar los pines 'N2HET' como pwm pero en nuestro caso no era posible. Esto es por culpa de que en la placa de salida para que funcione necesita una intensidad de corriente mínima de 6,4mA. Esto es debido a algunos componentes que tiene la placa de salida, que demandan esta corriente. Por desgracia no todos los pines del micro son capaces de abastecer con tanta intensidad de corriente. Los pines elegidos son capaces de proporcionar hasta 8mA por lo que no hay ningún problema y con estos pines nos aseguramos el correcto funcionamiento

del sistema.

Entradas Analógicas	4
Entradas Digitales	GIOB2, GIOB3, GIOA0, N2HET1[29], N2HET1[27], GIOA1, N2HET1[11], GIOA2
Salidas	EPWM1B, EPWM2B, EPWM3A, EPWM3B, EPWM4A, EPWM4B, EPWM7B, EPWM7A
Pantalla	MIBSPI5CLK, MIBSPI5SIMO[0], MIBSPI5SOMI[0], N2HET1[23], N2HET1[26]
UART	SCIRX, SCITX

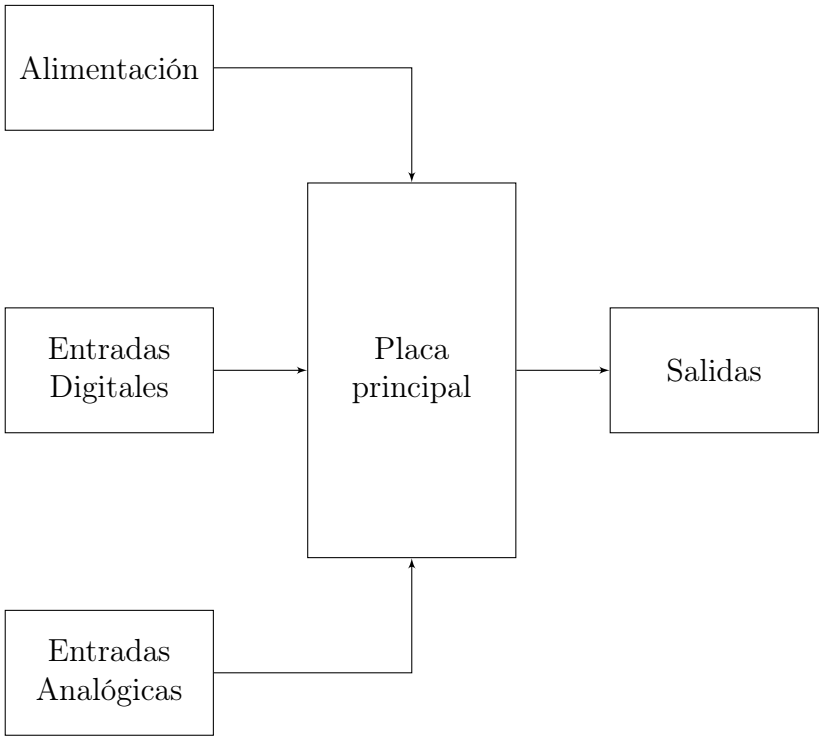
Cuadro 3.1: Pines del Hercules utilizados

El hecho de escoger el microcontrolador Hercules RM46x no ha sido simple casualidad. Este microcontrolador tiene la característica de estar diseñado para aplicaciones con una seguridad crítica. En nuestro caso estamos usando una placa de bajo coste que sirve como prueba para la línea de microcontroladores con gran seguridad. Su principal característica es que tiene un ARM Cortex-R4 de doble núcleo basado en la serie TMS570 MCU. Por ello es perfecto para este proyecto. La idea es crear un PLC mejorado y con este micro estamos garantizando que nuestras aplicaciones tendrán una seguridad crítica como podría ser una monitorización de gases. También tenemos la suerte de que Code Composer Studio (herramienta de edición y depuración) es compatible con el sistema que tiene nuestro micro. Es una suerte por el hecho de que desde HALCoGen se pueden exportar los códigos ha CCS. HALCoGen es una herramienta que nos permite configurar periféricos por medio de una interfaz gráfica. Esta configuración podemos convertirla a código que luego incluimos en nuestro proyecto en Code Composer Studio y podemos usarla sin ningún problema. Algunas características importantes de nuestro Hercules Launchpad son:

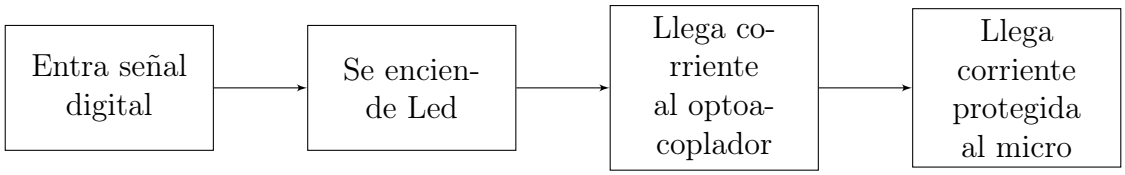
- Autotest incorporado para CPU y RAMs
- Módulo de señalización de errores con pin de error
- Monitoreo de voltaje y reloj
- Funcionamiento a 80 MHz
- Flash de 348kB con ECC (Error Correcting Code)
- 32kb RAM con ECC
- Flash de 16kB con ECC para emulación EEPROM
- Temporizador programable de 19 pines
- LIN/SCI(UART)
- Módulo de impulsos de codificador de cuadratura mejorado(eQEP)

Ahora vamos a ver un poco la estructura general de cada placa. Se presentarán en diagramas de bloques, solo para que se entienda la filosofía que se quiere seguir. En los próximos apartados se entrará en detalle sobre la distribución haciendo referencia a cada componente en concreto.

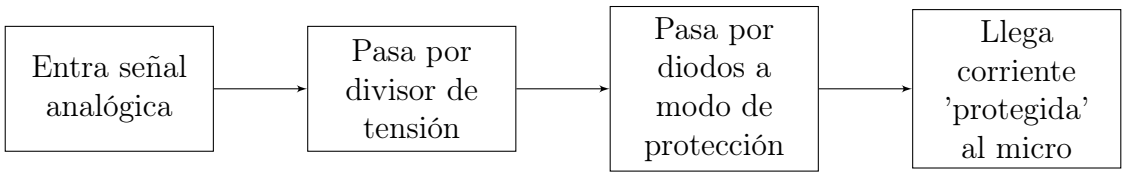
La placa principal no tiene un claro flujo de la corriente, por un lado entra alimentación al micro, entran también señales y salen otras señales.



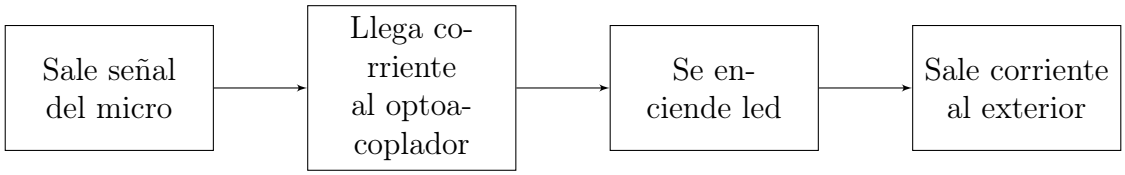
Para la placa de entradas digitales sería:



Para la placa de entradas analógicas:



Por último para las salidas sería:



Dentro de lo posible, se ha intentado colocar todos los componentes de tal forma que el microcontrolador quede protegido ante posibles fallos en el voltaje o la intensidad. Esto se ha conseguido haciendo una separación entre las señales que entran y salen directamente del microcontrolador con aquellas que entran y salen de nuestro sistema completo que simula un PLC. Así no hay riesgos de que llegue un voltaje más alto de lo que el microcontrolador pueda aguantar. Por desgracia no se ha conseguido un aislamiento completo debido a la placa de entradas analógicas. En cada apartado se explicará como se ha conseguido o porque no se ha conseguido en el caso anteriormente comentado.

Para terminar este apartado aquí se muestra un pequeño dibujo con la idea de como quede el sistema final con todas las placas conectadas entre ellas.

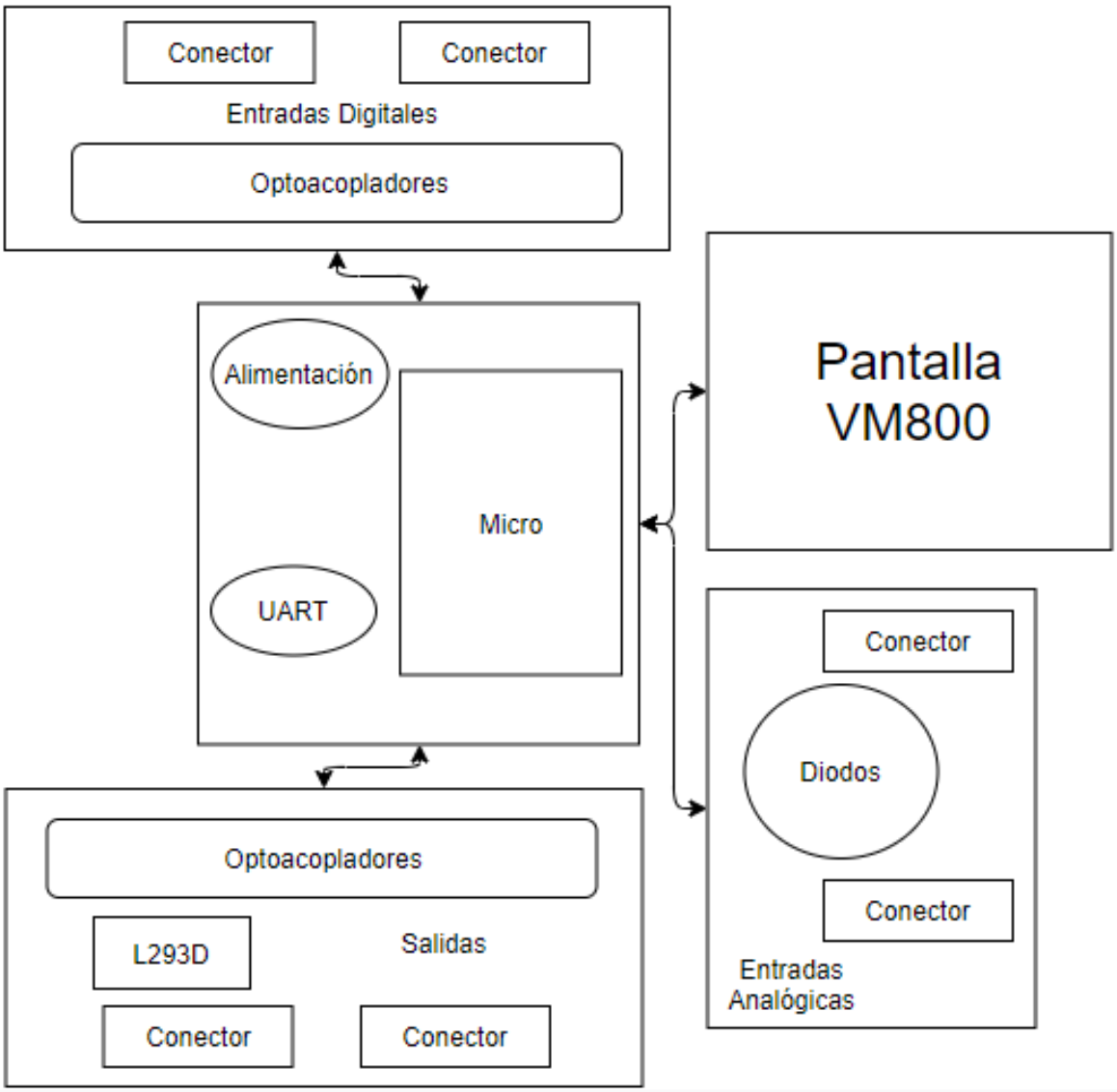


Figura 3.1: Diseño del proyecto completo

Capítulo 4

Diseño de placas electrónicas

En este apartado vamos a explicar con detalle todos los pasos que hemos seguido para diseñar el esquemático eléctrico de cada placa, todos los componentes que hemos usado y el diseño pcb de cada placa.

Para llevar a cabo el diseño de las placas, se ha usado el programa KiCad. Este programa es gratuito y puede usarse en cualquier sistema operativo. La ventaja que tiene frente a otros programas estudiados en la carrera es que no tiene ninguna restricción. Hay programas que poseen versiones gratuitas las cuales no te dan las mismas posibilidades que tienes al comprar dichos programas. Con KiCad no hay ningún problema al respecto.
El programa es bastante intuitivo y fácil de usar. En primer lugar, lo primero que tienes que hacer siempre es crear el esquemático de tu circuito. Al principio la idea era integrarlo todo en una placa. Todos los módulos tenerlos en una sola placa, cambiando así el modelo de proyecto que se ha visto en la introducción. Esta idea surgió del problema que había para crear dos esquemáticos distintos en un mismo proyecto, teniendo que incluir todos los circuitos en el mismo esquemático. La solución que se me ocurrió fue simplemente crear cuatro proyectos distintos, un proyecto para cada placa. El proyecto evolucionó de un modelo compacto a lo planteado en la introducción, un PLC Modular.
Teniendo cuatro proyectos separados, había que tener cuidado con el tema de las librerías. Cada componente de un esquemático necesita que se le asocie una huella. La huella es decir qué componente en la vida real se colocará en esa posición. Esto está ligado al diseño pcb, que es el paso final. Sin embargo, teniendo todos los proyectos en la misma carpeta junto con las librerías, no hay ningún problema.
Para el primer paso algo a destacar son las etiquetas en KiCad. Una forma muy útil de unir dos partes del circuito sin tener que llevar un cable muy largo que luego siempre puede incordiar con otros cables o componentes. Asignar la huella a cada componente es el segundo paso que hay que seguir. Una vez tengas todo el circuito montado, debes asignar a cada componente su huella para poder crear la Netlist. En algunos casos, puede que no encuentres la huella para el componente que necesitas ni siquiera buscando por internet alguna librería nueva. En ese caso, puedes crear tu mismo tu propia huella. En este proyecto, la huella que hace como conector entre el microcontrolador y la placa principal la he tenido que hacer por mi propia cuenta.

Por último, con la Netlist creada, podemos dar paso a diseñar la pcb. Simplemente tenemos que abrir una ventana nueva de diseño de pcb e importar la netlist que queramos. Con esto, conseguiremos que aparezcan todos los componentes en nuestra ventana con flechas de unión que marcan con que otro conector debe unirse cada conector de cada componente. A partir de ahí lo único que hay que hacer es empezar a enrutar e ir colocando las pistas hasta que quede un diseño compacto con todas las conexiones hechas. En este punto se mostrará gráficamente tanto los esquemáticos de los circuitos eléctricos como los diseños de las pcb donde aparecerán las huellas de los componentes.

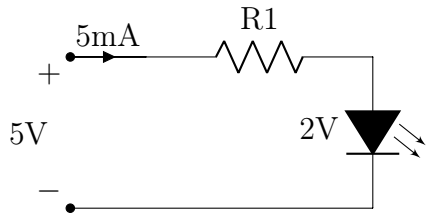
4.1. Placa principal

Es la placa que une a la CPU, el microcontrolador, y las placas de entradas y salidas. Aunque principalmente tiene los componentes necesarios para alimentar a todo el sistema, no es conveniente denominarla placa de alimentación porque existen otros componentes que mencionaremos a continuación que no tienen nada que ver mientras que siendo la placa de unión, parece más lógico llamarla placa principal.
En esta placa no se ve con claridad donde esta la separación de la que se hablaba en la pequeña introducción del apartado 2 para proteger al microcontrolador. La comentaremos con más detalle en los próximos apartados y si es oportuno se hará mención a la placa principal.

4.1.1. Componentes

La placa principal está compuesta por conectores para poder unirse a las placas de entradas/salidas, dos conectores que simulan la forma del microcontrolador que estamos usando, Hercules RM46x Launchpad, componentes necesarios para la alimentación del sistema y un integrado para poder conectarle al microcontrolador un bus de comunicaciones, el RS-485.
Para la parte de la alimentación se han usado condensadores para proteger los reguladores de voltaje. Teniendo en cuenta que queremos simular un PLC real, la placa principal se alimenta con un voltaje de 24V. Para el microcontrolador esta cantidad es demasiado grande y lo dañaría. Para que esto no pase, tenemos un regulador de voltaje que como salida proporciona 5V, ideal para su alimentación. Para comprobar que la alimentación de 5V esta llegando bien al microcontrolador, hemos colocado un led. Si ningún componente falla y la corriente llega bien, el led se iluminará cuando estemos alimentando al sistema, así siempre podremos comprobar si ha habido algún error en la primera parte del circuito. El led por supuesto tiene su correspondiente resistencia colocada en serie para disminuir la intensidad de corriente que le llega y estar seguro de que no lo quemamos.

Para los valores de las resistencias tenemos que tener en cuenta que hay que ajustarse a valores normalizados con una tolerancia del 10 %.
Los cálculos realizados para la resistencia usada como resistencia de limitación son los siguientes:



Utilizamos la ley de Ohm para calcular el valor de la resistencia R1.

$$V = R * I \tag{4.1}$$

$$R1 = \frac{5V - 2V}{5mA} = 600\Omega \tag{4.2}$$

Ajustando a los valores de resistencia normalizados:

$$R1 = 560\Omega \tag{4.3}$$

En la siguiente tabla se mostrarán cada uno de los componentes:

Componente	Número	Tipo
Conector 1x50	2	JST-1x50-P2.50mm
Conector 1x10	4	JST-1x10-P2.50mm
Conector 1x4	1	Phoenix-MSTB-1x4-P5.80mm
Conector 1x2	1	Phoenix-MSTB-1x2-P5.80mm
7805	1	Circuito integrado TO-220
LM117	1	Circuito integrado TO-220
max3079eapd	1	Circuito integrado 14DIP
Condensador	4	Capacitor-THT-D10.5mm
Resistencia	1	Resistor-THT-D2.5mm
Led	1	LED-THT-D2.00m

Cuadro 4.1: Componentes placa principal

En este apartado tenemos que hablar de otro componente, la pantalla VM800. En esta placa se encuentra un conector de 10 pines donde engancharemos la pantalla. Con ella la idea es darle una perspectiva más visual al proyecto, pudiendo seguir los pasos que se están realizando.
Aunque la pantalla tenga 10 pines, realmente no se usan todos estos pines. Esto quiere decir que habrá pines de nuestro conector que no están realmente conectados a ninguna otra parte. La pantalla pude alimentarse a 5V o a 3,3V, tiene 2 pines que se conectarían a tierra y el pin INT que solo sirve si se conecta a 3,3V. En nuestro caso, alimentaremos la pantalla a 5V por lo que dejamos esos 3 pines sin conectar a ningún lado (alimentación a 3,3V, una tierra y el pin INT).

Figura 4.1: Pantalla VM800

Es una pantalla gráfica con color, coprocesador gráfico, tiene un sistema de audio (sintetizador de audio basado en MIDI, amplificador y altavoz de 8Ω), es táctil y tiene una interfaz SPI con el host. El procesador es el FT800, que para la interfaz SPI trabaja hasta 30MHz. El cuello de botella de la pantalla es precisamente la interfaz con el usuario.
El modo de funcionamiento de la pantalla se basa en mandar por SPI los comandos o datos necesarios y leer de la memoria los datos de vuelta. El FT800 funciona en paralelo con nuestro MCU.

4.1.2. Esquemático eléctrico

A continuación se muestra una foto del esquemático seguido para posteriormente realizar el diseño de la pcb:

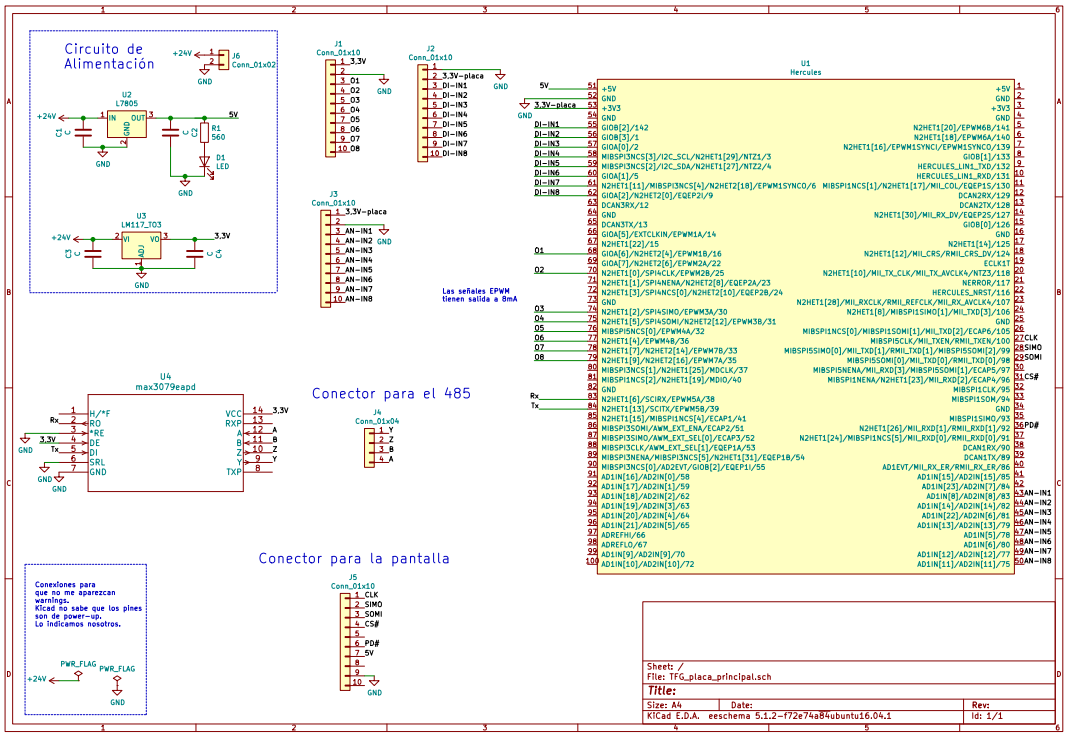


Figura 4.2: Esquemático de la placa principal

A partir de este esquemático se explicará la filosofía seguida para su diseño.

Existe un elemento grande que simula la forma del microcontrolador Hercules. Este esta compuesto por 100 pines. Para este proyecto solo hemos tenido que hacer uso de 34 de ellos. La distribución es la siguiente:

- 8 pines para las 8 entradas analógicas
- 8 pines para las 8 entradas digitales
- 8 pines para las 8 salidas
- 5 pines para controlar la pantalla led
- 2 pines para la comunicación UART
- 2 pines para la alimentación
- 1 pin para suministrar energía

Hagamos un repaso a la tabla de los componentes de la placa principal para ver como están todos relacionados entre ellos. Existen dos conectores de 50 pines, encargados de simular la forma del Hercules. A continuación, tenemos cuatro conectores de 10 pines cada uno. Uno de ellos es el que conecta con la pantalla. En este caso solo se usan 7 como se comentó en el apartado anterior. Los otros 3 son para las conexiones con las otras tres placas. Cada uno de ellos tiene en el conector los 8 pines que le corresponda para las ocho señales ya sean entradas o salidas y otros 2 pines más. Estos dos pines son el de tierra y un pin de alimentación. El pin de tierra es muy importante porque todas las tierras deben ser comunes y estar conectadas a la tierra del microcontrolador. El pin de alimentación se incluye porque en todas las placas existen componentes que necesitan energía. Sin embargo, no para todas las placas el pin de alimentación es el mismo. Esto tiene una explicación muy sencilla y volvemos a retomar el tema de proteger al microcontrolador. Para hacer una separación clara, hay algunos componentes que deben estar encerrados junto al microcontrolador como si de una burbuja se tratase. Para ello, en las placas donde los componentes que necesiten energía se encuentren dentro de esta zona protegida, el pin de alimentación es la misma alimentación que sale del Hercules. Estas placas son las de entrada, tanto las entradas analógicas como las entradas digitales. En la placa de las salidas, dichos componentes se encuentran fuera de la zona protegida, por lo tanto recibe una alimentación que viene del exterior. Esta viene de uno de los reguladores de voltaje que hay en la placa.
El regulador L7805, convierte la alimentación que le llega que son 24V a 5V que es el voltaje que necesita el microcontrolador para alimentarse. El otro regulador que hay en la placa es el LM117T. Este convierte el voltaje del exterior a 3,3V. Estos 3,3V son los que se conectan con la placa de las salidas para alimentar a sus componentes que han quedado fuera de la zona protegida. Estos 3,3V también se utilizan para alimentar al max3079eapd. A ambos lados de cada regulador hay un condensador conectado en paralelo, es decir, entre el voltaje y tierra. Estos condensadores se ponen para proteger un poco. A parte, en el L7805, existe un led previamente explicado.
Por último, existe otro componente en la placa, el integrado max3079eapd. Este integrado hace posible la conversión de UART a RS-485. Esta conectado a 2 pines del Hercules, al Rx y Tx. Estos pines son los que se utilizan para la comunicación UART. A través del integrado, somos capaces de recibir mensajes por el bus RS-485 y convertirlos de modo que el microcontrolador pueda recibirlos por el puerto UART y la operación inversa. Se ha incorporado esta pieza al proyecto por ser una parte común que suelen tener los PLC.
El bus RS-485 se caracteriza por transmitir a altas velocidades sobre largas distancias, sin importar si el medio es ruidoso. Esto se debe al par trenzado, que reduce mucho los ruidos que se puedan inducir. Tiene muchas aplicaciones distintas, puede usarse en sistemas de iluminación grande como en conciertos, en automatización, en plantas industriales, etc. Gracias a sus características comentadas y a la gran longitud que puede alcanzar este, tiene muchas ventajas.

4.1.3. Diseño pcb

En todas las pcb se siguen algunas reglas básicas que se explicarán en este apartado y no se volverán a repetir. Primero podemos contemplar la siguiente figura que servirá también de ejemplo para la explicación.

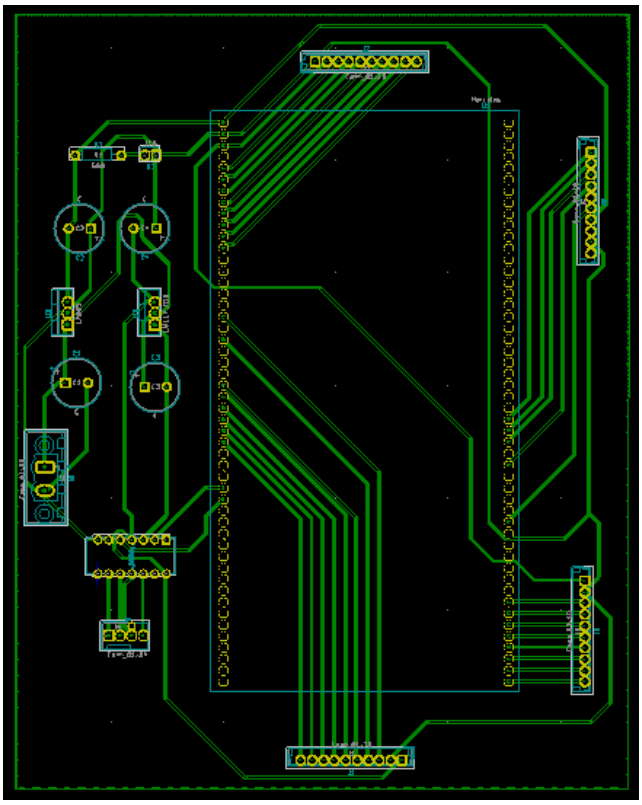


Figura 4.3: Diseño pcb de la placa principal

En todas las placas se siguen las reglas a continuación enumeradas, a menos que por alguna causa justificada no haya podido ser así.

- Las pistas no son más finas de 24 mils (0,6096mm)
- Las pistas no están unidas por ángulos rectos, sino ángulos de 135º
- Rellenar todo el área sobrante de la placa con tierra

De estas reglas básicas la única que no siempre se ha podido cumplir es la que tiene que ver con el tamaño de las pistas. Si nos fijamos un poco, en el diseño pcb de esta placa, hay veces que una pista cruza por en medio de lo que simularía al microcontrolador. Para ello, atraviesa entre 2 pines del microcontrolador. En estos casos, las pistas son algo más finas por temor a que pudiera causar algún cortocircuito o algún otro problema en el micro.
Como podemos ver en algunos casos las conexiones son un poco rebuscadas, teniendo que pasar entre medio de los conectores de algún componente. A parte del caso de los pines que se conectan al Hercules, se ve claramente como pasa en uno de los condensadores. Este enrutado algo más rebuscado se han hecho para poder hacerlo

todo en una única cara, evitando algunos inconvenientes. Si hiciéramos la placa a dos caras, algunos de los componentes tendrían que ir soldados por arriba. Esto tiene un claro inconveniente según que componente sea. Por ejemplo con los conectores que unen las diferentes placas es algo que no se puede hacer ya que no queda hueco entre el componente y la placa donde meter el soldador.
Sin embargo, no todas las placas se han podido hacer a una sola cara. Cada placa se explicará como se ha hecho en su apartado correspondiente.

En el apartado Resultados se mostrarán las placas verdaderas, cada componente y como queda todo finalmente soldado. Ahora solamente se quiere hacer conocer algunas peculiaridades que hayan podido surgir al hacer el diseño pcb y para ello nos apoyamos en una figura donde se pueden ver todas las pistas.

4.2. Placa Entradas Digitales

Esta placa simula un módulo de entrada del PLC. En este caso se trata de las entradas digitales. Esta placa tiene gran importancia por dos motivos: recibe las señales de entrada digitales que le llegarán al micro y protege al microcontrolador de posibles picos de tensión o intensidad que puedan producirse en estas entradas.
Para esta placa hay dos tipos de entrada distintas. La mitad de ellas son entradas a 3,3V y la otra mitad son a 24V. Se planteó la posibilidad de hacer que todas las entradas pudieran recibir señales del voltaje que fuese porque se controlaría con reguladores de tensión. El problema era que se necesitaban muchos reguladores por lo que finalmente se optó por obligar a que las entradas fueran de un voltaje específico. La mitad a 24V para simular entradas con un voltaje más real y la otra mitad a 3,3V para que sea todo lo contrario, hacer un sistema más pequeño que no admite voltajes tan elevados.

A continuación se explicarán todos los detalles de esta placa.

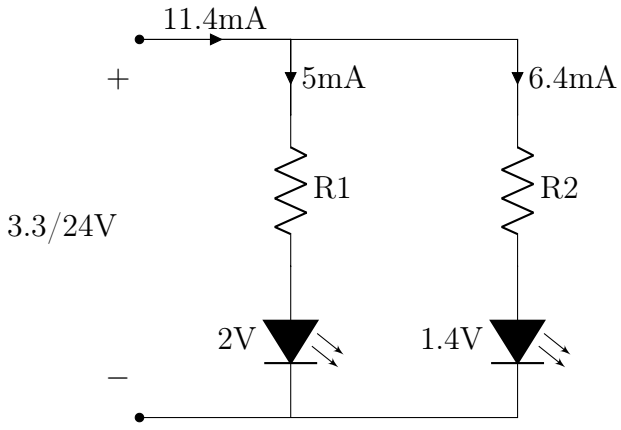
4.2.1. Componentes

La placa de entradas digitales está compuesta por dos conectores que se conectan con el exterior (por donde llegarían las señales) y un conector con el que se une a la placa principal. El conector de la placa principal es un conector de 10 pines donde se incluyen las ocho entradas digitales más la tierra y la alimentación de 3,3V que viene del Hercules. Los otros dos conectores son para recoger las señales del exterior. Cada uno recoge las señales de un determinado voltaje (24V o 3,3V). A parte, por cada dos señales hay una tierra. Esto se hace por varias razones. Si colocamos una sola tierra, al final tenemos muchos cables conectados al mismo punto y queda un poco feo. Si por el contrario ponemos una tierra para cada señal, tenemos conectores muy grandes y no es tan necesario. Poniendo una tierra cada dos entradas queda más limpio y no tenemos conectores tan grandes innecesariamente.

Esta placa tiene ocho optoacopladores. Los optoacopladores son la barrera que protege al micro de los posibles picos de voltaje. Este esta compuesto por un led

y una lógica de circuitos todo dentro de un circuito integrado. Su funcionamiento es sencillo, cuando la corriente llega al led, este emite luz infrarroja que activa el circuito que hay dentro del optoacoplador. Con el circuito activado, conseguimos tener una salida en nuestro optoacoplador.
Esa es una explicación un poco general, ahora entraremos más en detalle. Para empezar, con lo contado anteriormente, conseguimos la protección de la que tanto hemos hablado en los apartados anteriores. Normalmente, los optoacopladores se usa para proteger a circuitos eléctricos cuando entre él y el punto de entrada existe una diferencia de voltaje importante. En este caso, para las entradas a 24V, la diferencia de voltaje se ve bien clara. Para las entradas a 3,3V no es así pero se pone para que quede un circuito más compacto y protegido. Esta protección se consigue ya que toda la electricidad realmente se queda en el led que tiene dentro el optoacoplador. Solo se transmite la energía suficiente para encender todo el circuito que tiene detrás. Teniendo esta separación, vemos que hemos protegido al micro de cualquier pico de tensión que pudiera existir porque si por ejemplo pusiéramos una tensión mucho más alta de 24V, lo que pasaría es que probablemente se fundiría algún led pero la corriente se cortaría en el optoacoplador.
Una característica importante también es que tienen salidas invertidas. Es decir, el opto que usamos para el proyecto tiene una alimentación a 3,3V y su salida sin ninguna entrada será de 3,3V. Cuando le llegue corriente al led y este encienda todo el circuito, la tensión que nos devolverá el optoacoplador será de 0V. Esto es una característica común que suelen tener todos los tipos. Otra cuestión importante para los optoacopladores que vamos a usar, los 6N137, es que tienen la salida a colector abierto. Esto quiere decir que tenemos que poner obligatoriamente una resistencia de Pull-Up a la salida. Sin embargo, para las entradas esta resistencia no la pondremos física sino que configuraremos los pines de entrada del microcontrolador.

A parte, esta placa también tiene varias resistencias y varios led. Los led están colocados de tal forma que se enciendan cuando tenemos una señal de entrada por alguno de los conectores exteriores. Estos están en paralelo con los led internos que tienen los optoacopladores. A continuación se mostrará un circuito para que se vea como están dispuestos estos led con sus correspondientes resistencias de limitación y cuales han sido los cálculos que se han realizado para obtener el valor de las resistencias que hacen falta (recordemos que deben ser valores normalizados del 10 %).



Se ha dibujado únicamente un circuito porque es igual para ambos casos (entrada a 3,3V o 24V).
Utilizamos la ley de Ohm de nuevo (formula 4.1). Para las 4 entradas a 3,3V los valores de resistencias serían:

$$R1(3,3V) = \frac{3,3V - 2V}{5mA} = 260\Omega \quad (4.4) \qquad R1(3,3V) = 220\Omega \quad (4.5)$$

$$R2(3,3V) = \frac{3,3V - 1,4V}{6,4mA} = 296{,}875\Omega \quad (4.6) \qquad R2(3,3V) = 270\Omega \quad (4.7)$$

Para las entradas a 24V los valores de resistencia serían:

$$R1(24V) = \frac{24V - 2V}{5mA} = 4,4k\Omega \quad (4.8) \qquad R1(24V) = 3,9k\Omega \quad (4.9)$$

$$R2(24V) = \frac{24V - 1,4V}{6,4mA} = 3,531k\Omega \quad (4.10) \qquad R2(24V) = 3,3k\Omega \quad (4.11)$$

En la siguiente tabla se mostrarán cada uno de los componentes:

Componente	Número	Tipo
Conector 1x10	1	JST-1x10-P2.50mm
Conector 1x6	2	Phoenix-MSTB-1x6-P5.80mm
Optoacoplador	8	6N137
Resistencia	16	Resistor-THT-D2.5mm
Led	8	LED-THT-D2.00m

Cuadro 4.2: Componentes placa principal

4.2.2. Esquemático eléctrico

Para comenzar este apartado mostraremos la foto del esquemático eléctrico.

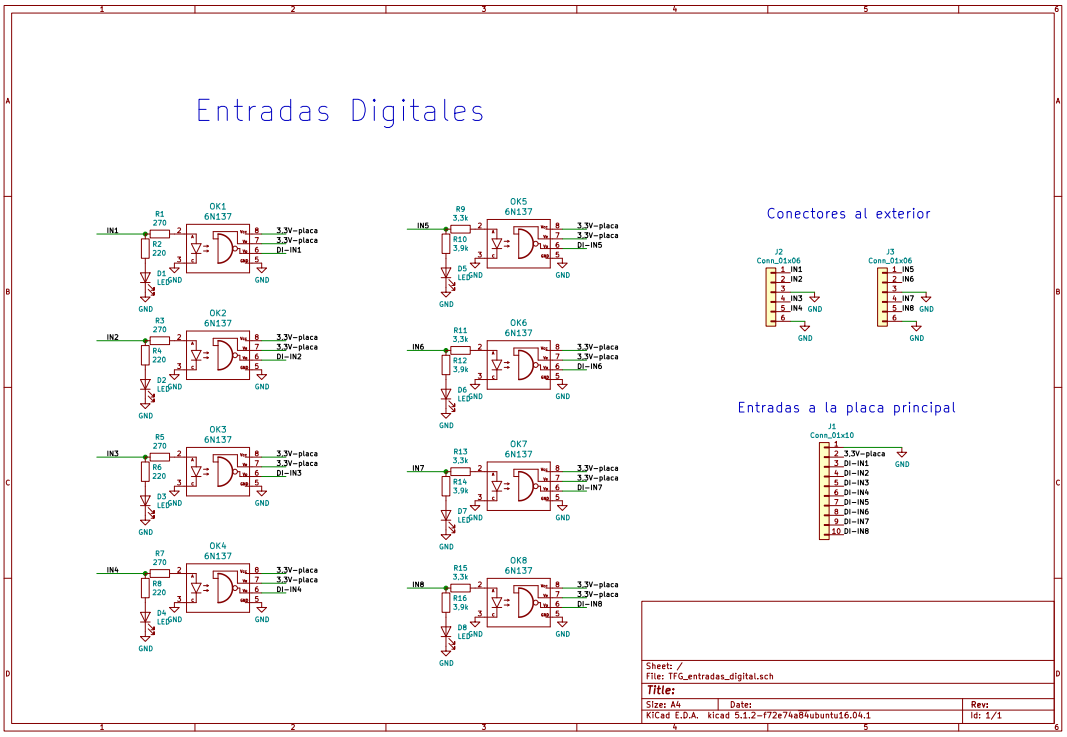
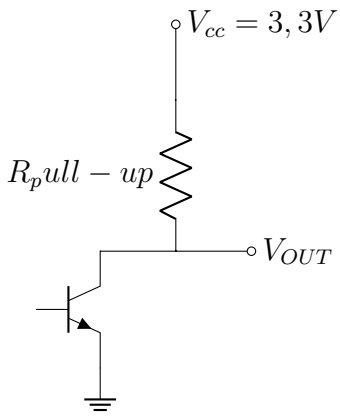


Figura 4.4: Esquemático de la placa entradas digitales

Vamos a comentar la distribución de esta placa. Los componentes más importantes son lógicamente los optoacopladores y todo el diseño eléctrico se construye en torno a ellos. Forman una clara barrera entre las conexiones con el exterior del circuito y el microcontrolador. Esta barrera se consigue gracias al funcionamiento de los propios optoacopladores pero hay un detalle muy importante que es determinante para que esto pueda ocurrir. Estos componentes se alimentan a 3,3V como se ha comentado en el apartado anterior, siendo determinante porque es el voltaje que obtendremos a la salida de los optoacopladores cuando estos no reciben señal ninguna a la entrada. Esta alimentación como se puede comprobar en la figura de arriba viene de la placa principal por el conector que une las dos placas. La diferencia es que no son los 3,3V que salen del regulador de tensión colocado en la placa principal en la parte del circuito de alimentación. Si fueran esos 3,3V, los optoacopladores se estarían alimentando con un voltaje que viene directamente desde el exterior. En ese caso, si se produjera cualquier pico inusual, afectaría directamente a la salida del optoacoplador y por lo tanto a la entrada del microcontrolador pudiendo dañarlo. Por ese motivo, la alimentación que reciben estos componentes viene directamente desde el micro. Hago esta aclaración dado que el nombre que recibe esta señal en el esquemático eléctrico es 3,3V-placa y puede traer alguna confusión. Realmente, si nos fijamos en la figura 4.2 puede verse claramente que esta tensión de 3,3V-placa

proviene directamente del micro, llamándose 3,3V la que proviene del regulador de tensión. Así aseguramos totalmente que los optoacopladores están creando una zona protegida para el microcontrolador.
Al inicio, el esquemático contaba también con resistencias de pull-up a la salida de los optoacopladores que como se ha comentado en el apartado anterior son necesarias al ser la salida a colector abierto. Esto se explica mejor con el siguiente esquema delante:



Aquí presentamos un diseño de un colector abierto. Para nosotros, el transistor que vemos formaría parte de nuestro optoacoplador, Vcc sería la alimentación que viene del microcontrolador hasta el optoacoplador y Vout es la señal que entra en el micro. El transistor en estos casos funciona como un interruptor. Cuando no activamos el transistor (en nuestro caso hablamos de que no le llega corriente al optoacoplador), la única conexión que tiene Vout es a Vcc, y cuando activamos el transistor (llega corriente a nuestro optoacoplador), Vout se conecta a tierra a través de este.
Este era parte del diseño principal del esquemático pero al final optamos por quitar esta resistencia física e introducirla en la configuración de las entradas del microcontrolador. El funcionamiento sería el mismo, y se ve claramente como con esta configuración tenemos una salida del optoacoplador que es inversa.

Por la zona no protegida del sistema, tenemos tanto las distintas resistencias como los led, como los conectores para que entren las señales digitales. De esta parte del circuito no hay mucho más que explicar que no hayamos comentado anteriormente. El diseño de como están colocados los led para identificar que ha entrado una señal ya sea de 3,3V o de 24V se ha explicado y dibujado en el apartado anterior con todos los valores de resistencias usados y el camino que sigue la corriente en ese caso. Por tanto finalizamos este apartado y empezamos el diseño de la pcb de esta placa.

4.2.3. Diseño pcb

Aquí mostramos el diseño pcb que hemos seguido para la placa de entradas analógicas:

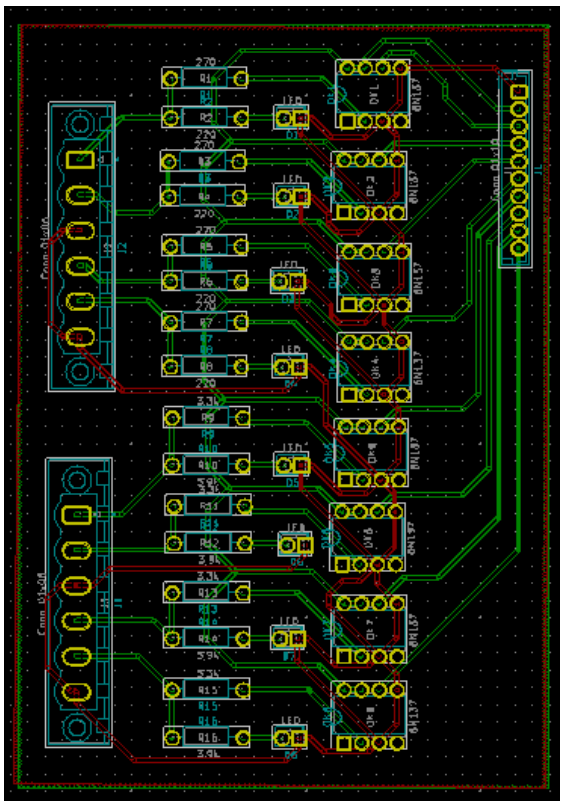


Figura 4.5: Diseño pcb de la placa entradas digitales

Como podemos comprobar, este diseño pcb se ha hecho a dos capas. Por un lado, tenemos la capa inferior (conexiones en verde), donde se hacen todas las conexiones necesarias menos las conexiones a tierra. Por otro lado, tenemos la capa superior (conexiones en rojo), donde se hacen las conexiones a tierra. Ambas caras tienen un relleno de toda el área sobrante con tierra. En este caso, la capa superior es toda un relleno de tierra a excepción de cada agujero que no este conectado a tierra. Se hizo así por la imposibilidad de conectar los distintos puntos de tierra en la capa inferior. Por culpa de los optoacopladores, los espacios entre pines de cada optoacoplador son muy pequeños y resulta imposible encontrar otra ruta alternativa. Hay zonas en las que algunos pines están totalmente encerrados por otras pistas y no es posible enlazarlos con otros. Haciendo la placa en dos capas solucionamos el problema y conseguimos que todo esté en orden.

En este diseño pcb se ha podido cumplir todas las reglas que se enumeraron en el apartado 4.1.3, no hemos tenido que recurrir a pistas más finas, todos los ángulos son de 135º y en ambas capas se ha rellenado el área sobrante con tierra.
Quizás se podría haber intentado colocar alguna pista más fina y no tener que hacer la placa en dos capas pero debían ser demasiado finas. Haberlo intentado

seguramente hubiera llevado a errores y tener que hacer de nuevo la placa por lo que se prefirió simplemente hacerlo en dos capas.

4.3. Placa Entradas Analógicas

Esta placa simula otro módulo de entrada del PLC, las entradas analógicas. Se podría haber hecho una sola placa (que simulara un único módulo), la cuál tuviera tanto entradas digitales como entradas analógicas. Sin embargo, había un problema en hacerlo así y era que las entradas analógicas no están aisladas galvánicamente. En ese caso, se crea un claro conflicto con las entradas digitales que si consiguen este aislamiento gracias a los optoacopladores.
Por eso mismo, se ha separado en dos placas distintas. Esto demuestra claramente la debilidad del diseño pero hablando con el tutor optamos finalmente por esta opción para no complicar tanto el diseño.

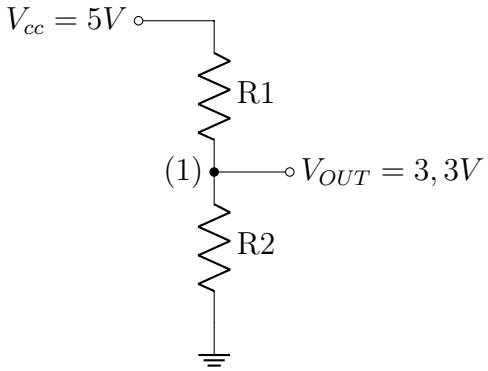
A continuación se explicarán todos los detalles de esta placa.

4.3.1. Componentes

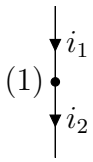
Esta placa está compuesta por los dos conectores encargados de recibir las señales de entrada (los que se conectan con el exterior) y el conector correspondiente para conectarse con la placa principal. Los conectores exteriores tienen seis entradas cada uno, cuatro para entradas que serán analógicas y dos a tierra. Lo mismo que en la placa de entradas digitales. El conector interior (el de unión con la placa principal), tiene 10 pines. Entre ellos se incluyen como es lógico todas las entradas analógicas que son ocho, un pin a tierra y otro a los 3,3V que proporciona la placa. Se usan el voltaje que genera la placa y no los 3,3V que podemos conseguir de la alimentación justamente por no estas aisladas galvánicamente.Aparte de los conectores, esta placa solo tiene resistencias y diodos.

La placa está preparada para recibir entradas analógicas de 5V. Esto quiere decir que todas las señales que estén en el rango entre 0V y 5V son bien recibidas. Si por algún casual alguna entrada superara este voltaje podríamos dañar el microcontrolador.
Aún así, las entradas del microcontrolador aceptan como máximo 3,3V mientras que nosotros estamos aceptando señales de hasta 5V. Esto es gracias al divisor resistivo que tenemos puesto en cada entrada. Con él las señales que como entrada están en el rango entre [0-5]V, llegarán al microcontrolador entre [0-3,3]V.

Con el siguiente dibujo se mostrarán los cálculos realizados para escoger los valores de estas resistencias.



Para los cálculos de estas dos resistencias se ha utilizado la ley de las corrientes de Kirchhoff. En concreto la primera ley de Kirchhoff o también conocida como ley de nodos. Esta ley dice que la suma de las corrientes que entran en un nodo debe ser igual a la suma de las corrientes que salen de ese nodo. Para nuestro caso, escogemos el nodo que se ve señalado en el dibujo anterior. Para ese nodo tenemos tres corrientes y nosotros las pondremos de la siguiente manera:



La primera ley de Kirchhoff dice que:

$$i1 = i2 \tag{4.12}$$

$$\frac{Vcc - Vout}{R1} = \frac{Vout - GND}{R2} \tag{4.13}$$

Despejando Vout finalmente obtenemos:

$$Vout = \frac{R1}{R1 + R2} * Vcc \tag{4.14}$$

Sustituimos Vout y Vcc por los valores conocidos y elegimos los valores de las resistencias según la relación entre ellas y sabiendo que deben ser valores normalizados del 10 %.

$$3,3 = \frac{R2}{R1 + R2} * 5 \tag{4.15}$$

$$3,3 * R1 = 1,7 * R2 \tag{4.16}$$

$$R1 = 27k\Omega \tag{4.17}$$

$$R2 = 47k\Omega \tag{4.18}$$

Se escogen valores de resistencias elevados para demandar menos corriente. Con los siguientes valores realmente estamos acotando aún más el rango (por culpa de los valores normalizados de resistencia). Está entre [0-3,18]V.

Los componentes de esta placa son:

Componente	Número	Tipo
Conector 1x10	1	JST-1x50-P2.50mm
Conector 1x6	2	Phoenix-MSTB-1x6-P5.80mm
Resistencia	16	Resistor-THT-D2.5mm
Diodo	16	Diode-THT

Cuadro 4.3: Componentes placa principal

4.3.2. Esquemático eléctrico

Aquí podemos ver una foto del esquemático eléctrico.

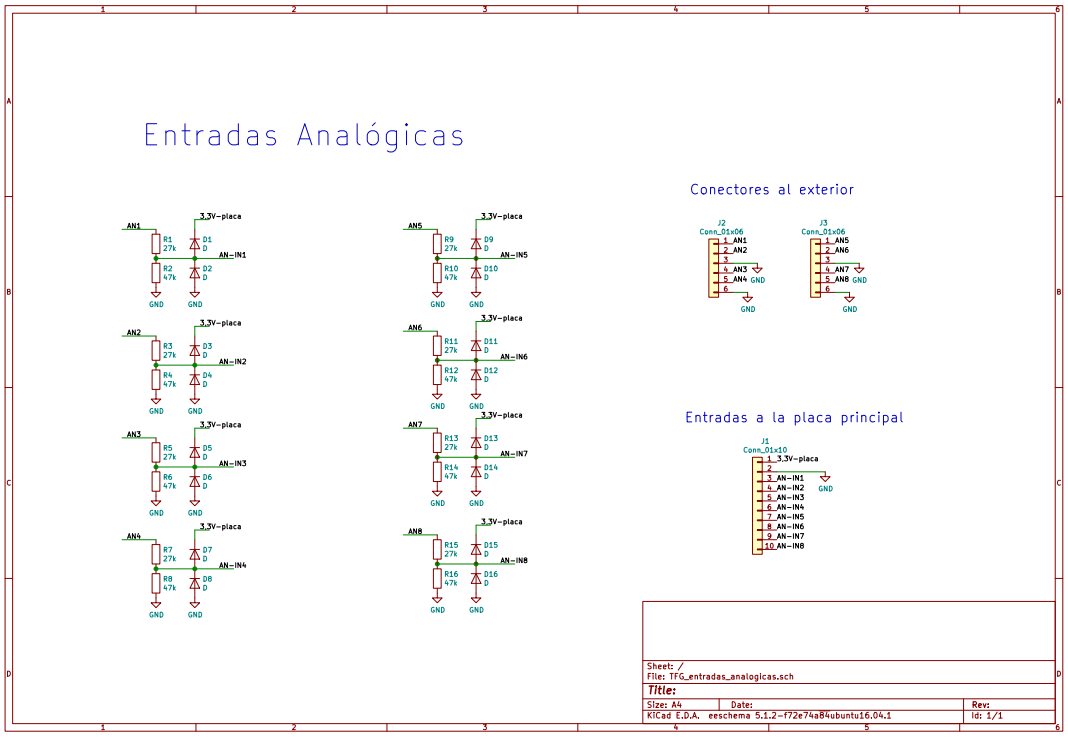


Figura 4.6: Esquemático de la placa entradas analógicas

El esquema eléctrico que se ha diseñado para esta placa no tiene mucho que comentar. Simplemente se conectan las entradas analógicas al divisor resistivo explicado en el apartado exterior, y la salida de ese divisor resistivo se conecta directamente a la conexión que tiene la placa con el microcontrolador.
En el apartado de componentes de esta placa no hemos hablado de los diodos, simplemente los hemos documentado en la tabla correspondiente. Los diodos como componente no tienen mucho más que decir por eso me parecía oportuno explicar

su uso y como están dispuestos en este apartado.
Como se puede comprobar en la figura 4.6, los diodos están puestos después del divisor resistivo. El primer diodo esta conectado desde tierra hasta la salida del divisor resistivo. El segundo diodo esta conectado desde esta misma salida hasta la toma con la alimentación de 3,3V que vienen del microcontrolador. Dado que no hemos usado ningún componente para aislar galvánicamente estas entradas, hemos decidido colocar estos diodos a modo de pequeña seguridad. Los diodos en este caso funcionarán como interruptores, podrán conducir corriente o comportarse como circuito abierto. Como ya sabemos, los diodos conducen a partir de un cierto voltaje si el ánodo esta conectado al punto positivo y el cátodo al negativo. Esto es lo mismo que si decimos que entre el ánodo y el cátodo tiene que haber una diferencia de potencial positiva. Para el primer diodo que hemos definido, si no hay alteraciones en la tensión de salida de nuestro divisor resistivo, la diferencia de potencial que tendría sería entre GND (0V) y el voltaje en la salida del divisor (entre 0 y 3,3V). Esta diferencia sería por tanto de 0V o tendría valor negativo y el diodo no conduciría, comportándose como un circuito abierto. En caso de que hubiera alguna alteración y tuviéramos tensión negativa a la salida del divisor, entonces el diodo conduciría. El diodo recortará el exceso de tensión que podría dañar a nuestro microcontrolador.
Para el segundo diodo, el funcionamiento es el mismo pero con otros valores. En este caso, el ánodo del diodo se encuentra entre 0 y 3,3V y el cátodo está siempre a 3,3V. Si no hay ningún fallo y la tensión a la salida del divisor de tensión no supera los 3,3V, el diodo no conducirá. Si por el contrario ocurre cualquier fallo y esta tensión se incrementa, el diodo empezará a conducir y de nuevo recortará un poco ese exceso de tensión sufrido. Con este montaje, no estamos protegiendo al microcontrolador contra cualquier perturbación seria que pueda sufrir pero si podemos asegurar que si hay pequeñas variaciones de tensión entre los límites establecidos, los diodos nos ayudarán a compensar ese exceso de tensión que puede llegar a dañar nuestro sistema.

4.3.3. Diseño pcb

A continuación se muestra el diseño pcb que se ha implementado para esta placa.

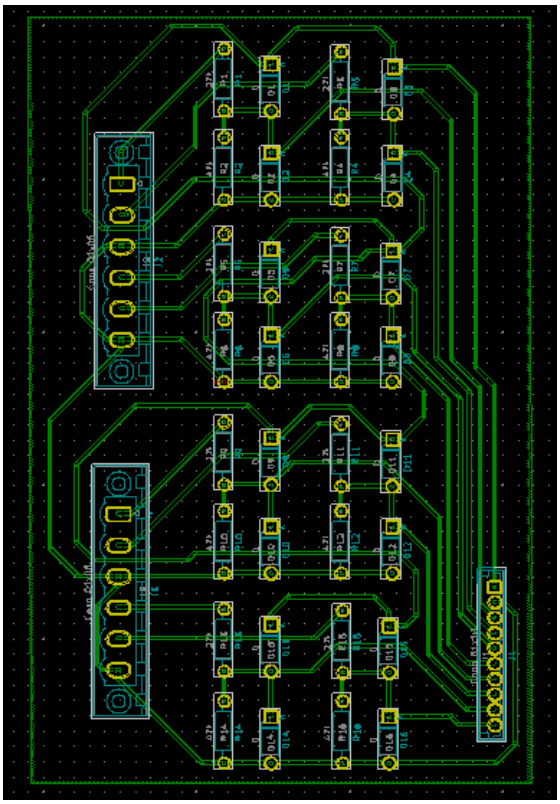


Figura 4.7: Diseño pcb de la placa entradas analógicas

Esta placa está hecha solo sobre la capa inferior. Al no tener componentes que sean circuitos integrados no hay problemas con las conexiones. La mayoría de las veces los inconvenientes vienen por componentes como los circuitos integrados, los cuales tienen sus patas muy juntas y no hay espacio para colocar una pista entre ellas. Por eso deja pocas vías por donde construir tu diseño. En esta placa, al tener únicamente resistencias y diodos, podemos poner pistas que atraviesen estos componentes y no tenemos problemas para unir ningún punto. Esta vez se ha podido cumplir todas las reglas generales que estamos usando: pistas no más finas de 24mils, ángulos de 135º y un relleno de toda la parte inferior (donde estamos colocando las pistas) conectado a tierra.
Probablemente el diseño más fácil y por lo tanto no tiene mucho que comentar así que pasamos al siguiente apartado.

4.4. Placa salidas

Esta placa simula un módulo de salidas del PLC. En el apartado 3, Arquitectura general de diseño, se explicó el porqué solo hay una placa de salidas y no dos (una para salidas analógicas y otra para salidas digitales) como sería lógico.

Sin embargo, al igual que la placa de entradas digitales, está bien protegida gracias al uso de optoacopladores. Por otro lado, también se distingue entre salidas a 3,3V y salidas a 24V para que sean compatibles con el diseño de las entradas digitales.

Veamos los detalles de esta placa.

4.4.1. Componentes

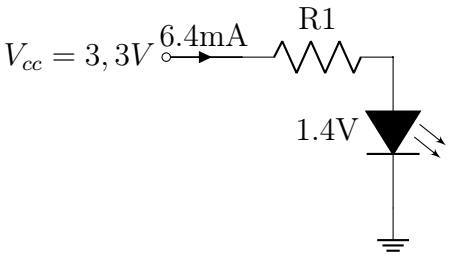
La placa de salidas esta compuesta por un conector con 10 pines para comunicarse con la placa central y otros dos conectores de 8 pines cada una para conectarse con el exterior del sistema. El conector que se comunica con la placa central está compuesto por las ocho salidas del microcontrolador, la tierra y una alimentación de 3,3V que viene de la parte de alimentación, es decir, del regulador de tensión. De los otros dos conectores, uno se usa para conectar las salidas que queramos a 24V y el otro para las salidas a 3,3V. Cada conector tiene cuatro salidas conectadas. A parte de estas cuatro salidas, por cada dos, existen tanto una tierra como una alimentación que será la correspondiente en cada caso (3,3 o 24V según el voltaje de esas salidas).

Esta placa también tiene ocho optoacopladores. Los optoacopladores son del mismo tipo que los usados para la placa de entradas digitales por lo que funcionan de la misma manera. Por eso, no se explicará de nuevo el funcionamiento y solo se tendrá en cuenta las explicaciones necesarias para argumentar el esquemático eléctrico general que por supuesto se abordará en el siguiente apartado.

Un componente nuevo que tenemos en esta placa es el L293D. Es un circuito integrado el cuál por dentro tiene cuatro circuitos distintos para manejar cuatro entradas diferentes. La característica que ha sido determinante para usar este driver ha sido que tiene la posibilidad de utilizar dos tensiones diferentes. Una tensión será la de alimentación para el circuito integrado y la otra para alimentar lo que queramos a la salida de este. En esta placa necesitábamos algo similar dado que cuatro de las ocho salidas tienen que ser a 24V, mientras que lo normal para las señales de salida pasando por el optoacoplador sería tener una tensión de 3,3V. Con este integrado somos capaces de aumentar la tensión y poder trabajar con salidas a 24V (el integrado acepta voltajes entre 4,5V y 36V y tiene la capacidad de controlar corrientes de hasta 600mA).

Por último, esta placa es la que más resistencias tiene, vamos a ver como están dispuestas cada una para poder demostrar como se ha calculado el valor de cada una de ellas.

Para empezar, a la salida del microcontrolador tenemos 3,3V y necesitamos al menos 6,4mA de corriente para que el optoacoplador empiece a funcionar (mismas condiciones que los optoacopladores de la placa de entradas digitales). Para ello tenemos la siguiente resistencia:



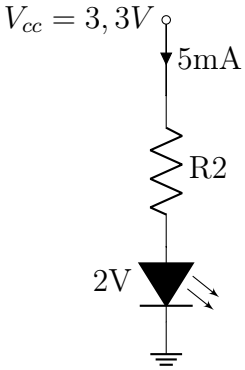
Utilizando la ley de Ohm (formula 4.1):

$$R1 = \frac{3,3V - 1,4V}{6,4mA} = 296,875\Omega \tag{4.19}$$

Ajustando a valores normalizados del 10 %:

$$R1 = 270\Omega \tag{4.20}$$

En la salida del optoacoplador, hemos colocado el led con su correspondiente resistencia de limitación. Como sabemos los optoacopladores devuelven valores invertidos. Los led están colocados de tal forma que se enciendan cuando en la salida del microcontrolador pongamos un 1, es decir, 3,3V, y que por lo tanto a la salida del optoacoplador tengamos un 0. A continuación se muestra como están colocados y como se ha calculado la correspondiente resistencia de limitación.



Este dibujo representa el caso en el que la salida del optoacoplador este a 0 y se simboliza en el dibujo con la tierra. En este caso, las cuentas realizadas son las siguientes:

$$R2 = \frac{3,3V - 2V}{5mA} = 260\Omega \tag{4.21}$$

Ajustando a valores normalizados del 10 %:

$$R1 = 220\Omega \tag{4.22}$$

A parte de estas dos resistencias, también existe una resistencia más que tiene que ponerse en paralelo con el led y su resistencia de limitación. Es una resistencia de pull-up y le damos un valor elevado para que no pase mucha corriente. El valor más común para este tipo de resistencias es de 10kΩ.

El listado con todos los componentes es el siguiente:

Componente	Número	Tipo
Conector 1x10	1	JST-1x50-P2.50mm
Conector 1x8	2	Phoenix-MSTB-1x6-P5.80mm
Optoacoplador	8	6N137
L293D	1	Circuito integrado
Resistencia	24	Resistor-THT-D2.5mm
Led	8	LED-THT-D2.00m

Cuadro 4.4: Componentes placa salidas

4.4.2. Esquemático eléctrico

Aquí tenemos una figura con el esquemático eléctrico que se ha seguido para esta placa.

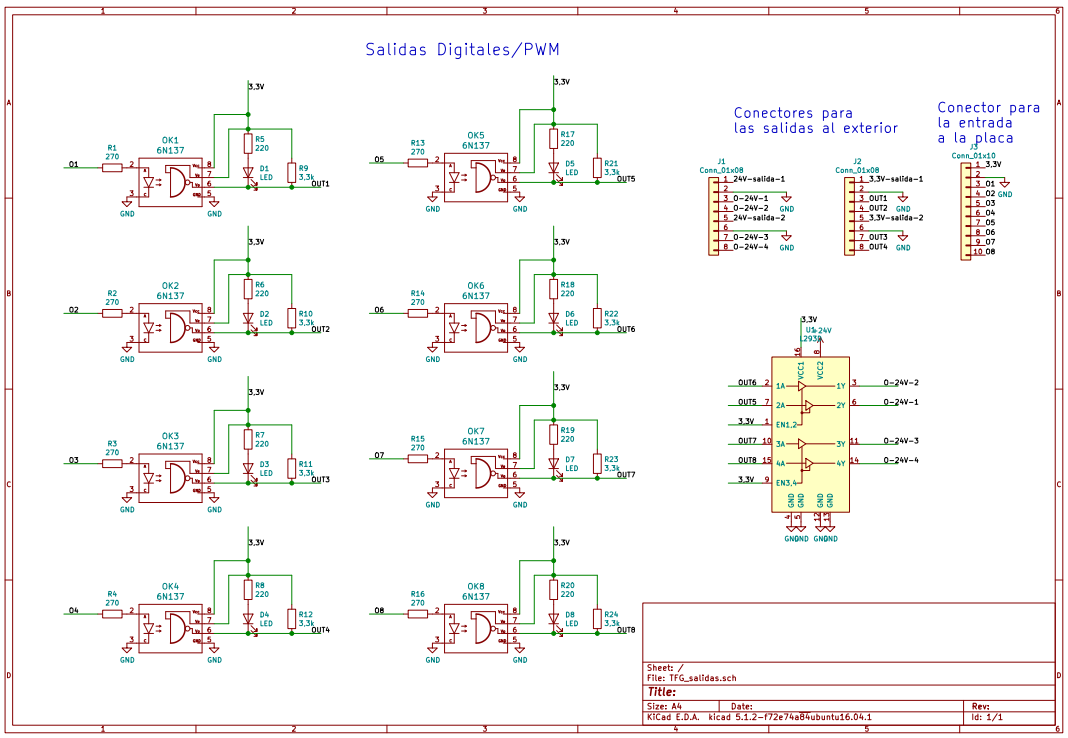


Figura 4.8: Esquemático de la placa salidas

Al igual que para la placa de las entradas digitales, en esta placa los componentes más importantes son los optoacopladores por garantizarnos la seguridad del microcontrolador en el sistema. Siguen la misma filosofía que en la placa de entradas digitales pero como es lógico el diseño es distinto. Como ya sabemos, la corriente eléctrica que necesitan estos componentes para activarse es como mínimo de 6,4mA. Para ello, se han escogido salidas del microcontrolador que son capaces de dar hasta 8mA, no teniendo así ningún problema en llegar al mínimo que necesitan. Lo único que hemos tenido que hacer es colocar una resistencia de limitación entre la salida del microcontrolador y los optoacopladores, cuyo valor se ha calculado en el apartado anterior. Para la salida de los optoacopladores, es necesario explicar como se han hecho las conexiones correspondientes.
Para empezar, se ha colocado un led para poder comprobar de forma visual cuando estamos encendiendo una salida. La cuestión es que realmente cuando el led se encienda tendremos 0V en esa salida del optoacoplador debido a que estos devuelven valores invertidos como se ha comentado anteriormente. Esto puede provocar al principio un poco de confusión al pensar que el aparato que queramos accionar debe hacerlo gracias al voltaje que le proporcionemos a la salida pero tiene fácil respuesta. Realmente, nosotros estaremos alimentando al aparato correspondiente siempre con 3,3V o 24V (tenemos las dos opciones) y por la otra entrada al aparato vendrá la señal de salida que estamos generando con nuestro sistema. Así, si nuestro microcontrolador no esta activando ninguna salida, tendremos un voltaje alto a la salida de nuestros optoacopladores. Esto haría que el aparato viera por ambos lados a los que se conecta la misma tensión. Sin embargo, cuando activemos la salida del microcontrolador, tendremos 0V a la salida de nuestros optoacopladores y los aparatos verían una diferencia de potencial entre los 2 puntos a los que se conectan por lo que se activarían. Es una lógica inversa que debemos usar debido a la salida inversa que nos proporcionan estos aparatos.

Para conseguir las salidas a 24V lo único que hemos hecho es colocar el integrado L293D detrás de cuatro de las salidas que tenemos. El integrado aumenta el voltaje que obtenemos a la salida de los optoacopladores igualándose al voltaje con el que alimentaremos a los aparatos que queramos conectar a su salida.
Se puede ver claramente en el esquema como los conectores que dan al exterior tienen dos tomas por las que se conectarían los aparatos para conseguir este voltaje del que estamos hablando para que puedan activarse. Realmente, estos pines no están conectados a ningún otro sitio de la placa. La idea es conectar en ese mismo pin un alimentación de 24V o 3,3V según de que conector estemos hablando y los cables necesarios para que les llegue ese voltaje a los aparatos de salida. Es un punto de unión en el que colindarán tres cables como máximo lo que establecimos que estaba bien y no era demasiado.

Otro tema a explicar es el voltaje de alimentación que les llega a los optoacopladores. Como en la placa de entradas digitales, tienen que alimentarse con 3,3V pero esta vez esos 3,3V no vienen del microcontrolador. Esta alimentación procede de los 3,3V que conseguimos gracias al regulador de tensión que se encuentra en la placa principal. Esto es debido a que los optoacopladores se alimentan por la parte

de su salida. En este caso, su salida no corresponde a la zona protegida del micro, sino todo lo contrario. La zona protegida acaba justo cuando la corriente para por dentro del optoacoplador. Por eso, se alimentan con los 3,3V que vienen del exterior. Así conseguimos mantener una separación entre la parte del sistema protegida y la parte del sistema que tiene contacto con el exterior siempre.

4.4.3. Diseño pcb

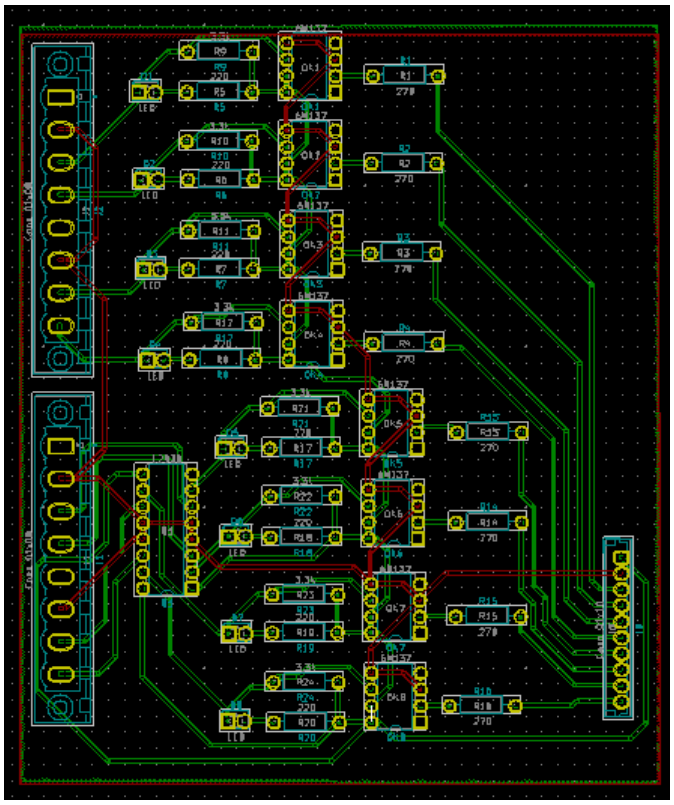


Figura 4.9: Diseño pcb de la placa salidas

Esta última placa es bastante parecida a la placa de entradas digitales por lo que también esta hecha a dos capas. Esto pasa por la misma razón, por culpa de los optoacopladores. Después de mucho intentarlo, es imposible conectar todos los pines entre ellos por lo que recurrimos a doble capa para solucionar el problema.
La capa de abajo es la que podemos ver en la siguiente figura con las conexiones verdes. En ella se conectan todos los pines menos los pines de tierra. Sin embargo, el relleno que tiene se hace con tierra como siempre hemos dicho y cumpliendo con la regla que comentamos al principio. La capa superior, contiene las conexiones entre los puntos de tierra que quedan aislados en la capa inferior y se aprecia en la figura con las pistas en rojo.

Aparte de estos detalles, no hay mucho más que decir. Se han cumplido todas las reglas que hemos puesto para los diseños de las cuatro pcb sin mayor complicación. Es una placa un poco más grande debido a que es la que más componentes

tiene y para que quedara un diseño más claro se ha intentado juntar lo máximo los componentes sin llegar a forzar demasiado para evitar errores.

4.5. Placa auxiliar

Esta placa no aparece con anterioridad comentada debido a que no es una parte más dentro de las funcionalidades que puede llegar a tener nuestro PLC. Esta placa ha sido simplemente creada a raíz de un fallo encontrado en la huella de KiCad con uno de los componentes. Este componente es el LM117. El componente usado en KiCad para obtener su huella y poder hacer la placa principal con el correspondiente circuito estaba equivocada. Me dí cuenta una vez quise soldar el componente ya que la distribución de las pistas me parecía algo extraño. Cuando revisé como debía ser el componente me encontré con la sorpresa de que la distribución de los pines era distinta a la que por defecto aparecía en el programa. Además tuve que añadir un par de resistencias por lo que decidimos crear una placa muy pequeña que pudiera solventar todos estos inconvenientes de una sola vez.
Debido a la simpleza de la placa este apartado será muy breve y apenas tendrá contenido.

4.5.1. Componentes

Esta placa solo tiene tres componentes, dos resistencias y el LM117. Aparte tiene tres agujeros que es donde se suelda la placa a la placa principal. Para calcular el valor requerido de las resistencias se siguió la siguiente fórmula, la cuál se puede encontrar en el datasheet del LM117.

$$Vout = 1,25V(1 + \frac{R2}{R1}) + Iadj(R2) \tag{4.23}$$

Despreciando la intensidad para nuestro caso y sabiendo que el voltaje de salida que nosotros queremos es de 3,3V la cuenta es muy fácil.

$$R2 = 1,64 * R1 \tag{4.24}$$

$$R1 = 220\Omega \tag{4.25}$$

$$R2 = 330\Omega \tag{4.26}$$

Componente	Número	Tipo
Resistencia	2	Resistor-THT-D2.5mm
LM117	1	Circuito integrado TO-220

Cuadro 4.5: Componentes de la placa auxiliar

4.5.2. Esquemático eléctrico

Realmente las conexiones en KiCad son muy pocas y no queda claro como deben disponerse los componentes por ello he puesto una fotografía sacada del datasheet del componente donde podemos ver claramente como debe ir conectado el componente.

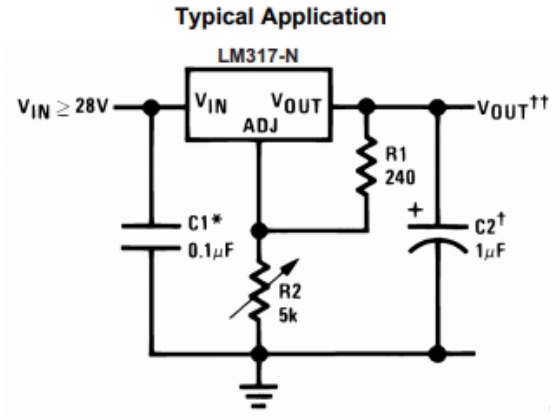


Figura 4.10: Esquemático placa auxiliar

En este caso, los condensadores sí están en la placa principal dado que estaban así desde un principio.

4.5.3. Diseño pcb

Aquí podemos ver una imagen con el diseño pcb de esta placa.

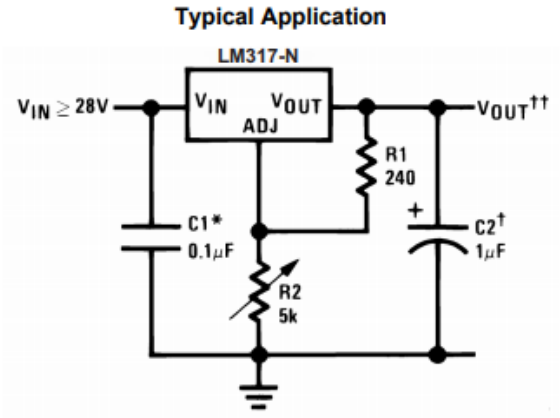


Figura 4.11: Diseño pcb de la placa auxiliar

La única novedad que hay sobre esta placa es el hecho de que se ha soldado por arriba. Debido a que ninguno de los componentes era un inconveniente para esta práctica, se decidió soldar por arriba para que fuera más fácil soldarla directamente a la placa principal sin necesidad de usar ningún conector.

En el apartado de resultados donde aparecerán las distintas fotografías de las placas reales podremos ver como ha quedado finalmente esta placa unida a la placa principal.

Capítulo 5

Programación del microcontrolador

La programación de este sistema se ha llevado a cabo sobre Code Composer Studio, un entorno de desarrollo integrado (IDE) para desarrollar aplicaciones para procesadores de Texas Instruments. Nuestro microcontrolador, el Hercules, es de la empresa Texas Instruments por lo que hemos aprovechado este entorno de desarrollo. También por ser con el que durante varios cursos hemos trabajado (siempre usando microcontroladores de Texas Instruments). A parte, también hemos usado una aplicación, HALCoGen, que permite a los usuarios generar controladores de dispositivos HAL para Hercules. Se denomina HAL a la capa de abstracción de hardware (siendo HAL sus siglas en inglés). Con HALCoGen lo que hemos hecho es preparar las librerías que normalmente tendríamos que hacer a mano o coger de internet de una manera muy simple y dinámica. Con una interfaz gráfica, puedes escoger los periféricos que vas a usar del micro, y activar los pines que creas necesario o configurar algunos puertos más específicos. Cuando has terminado toda esta configuración, se generan de forma automática todos los archivos que servirán para usarlos como librerías en el código a desarrollar.

La programación de nuestro sistema debe controlar las señales de entradas que le llegan, tanto digitales como analógicas, debe accionar las diferentes salidas, que serán por medio de modulación por ancho de pulsos (PWM, siglas en inglés de pulse-width modulation), y controlar la pantalla que conectaremos a la placa principal. Para ello los periféricos que hemos tenido que usar son los siguientes:

	Periférico
Entradas analógicas	ADC1 driver
Entradas digitales	GIO driver y HET1 driver
Salidas	ETPWM driver
Pantalla	MIBSPI5 driver

Cuadro 5.1: Periféricos usados del microcontrolador

Con la pantalla nuestro objetivo es crear una pequeña de interfaz gráfica donde se pueda ver como se va interactuando con el sistema. Para ello lo que representaremos será ocho pequeños círculos que se encenderán y apagarán en función de si recibe o no una señal digital a la entrada. Será lo mismo que los leds físicos que hemos colocado

en la placa correspondiente. También aparecerán ocho números que representarán los voltios que están entrando por las señales analógicas. Para esto haremos una conversión sabiendo que los bits que posee el ADC driver son 12. Por último, en la pantalla aparecerán ocho botones que podremos pulsar para accionar las salidas que tenemos.

Un dibujo de como quedaría la pantalla sería más o menos así:

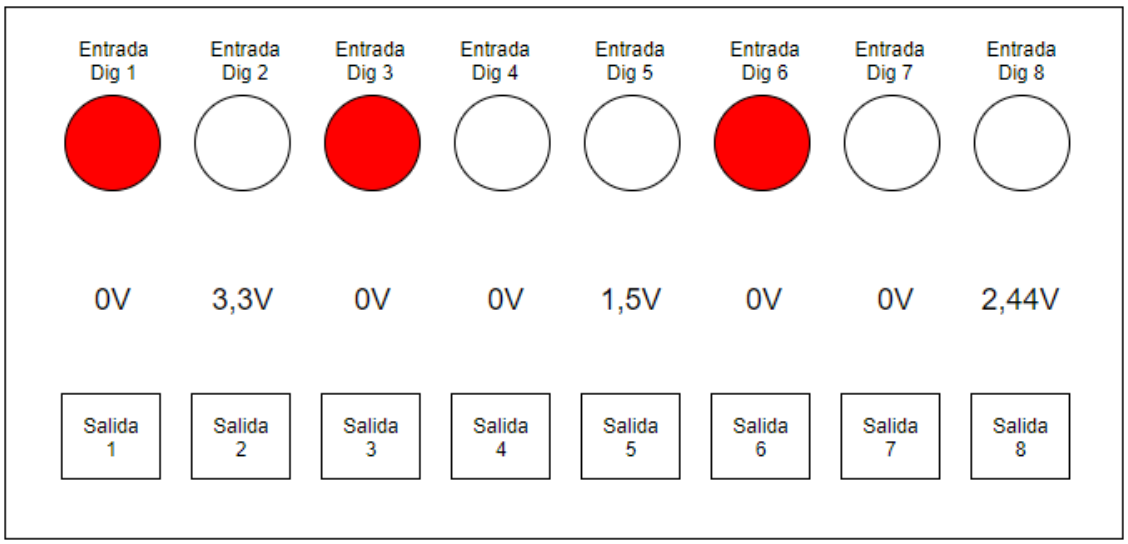


Figura 5.1: Ejemplo de la pantalla

Para ello configuraremos primeramente una pantalla inicial en la que todos los círculos aparezcan sin color, es decir, apagados, y todos los valores con los voltios aparezcan a 0V. Después, con cada evento que se produzca en el sistema, actualizaremos la pantalla según que haya ocurrido. Para explicar esto un poco más es mejor pasar a desarrollar directamente la filosofía del código que se ha utilizado para este sistema.

El PLC es un sistema en tiempo real y como tal lo he programado para que actúe frente a eventos. Esto que quiere decir, que el código no está hecho de forma lineal y que no seguirá siempre unos mismos pasos dentro de un bucle o algo parecido. En nuestro caso, al principio del programa iniciaremos todas las variables y librerías de las que luego queremos emplear. Lo siguiente será crear las tareas que se desarrollarán todas en paralelo y en tiempo real. Por último, se activan todas las tareas previamente programadas y el sistema empezará a reaccionar frente a los distintos eventos que tengan lugar en las distintas tareas creadas.
El código consta de cuatro tareas distintas:

- Si detectan una entrada digital, pone a 1 la variable asignada a esa entrada
- Si detectan una entrada analógica, pone a 1 la variable asignada a esa entrada y guarda el valor que le haya llegado
- Si detectan que alguna variable esta a 1, activan la salida correspondiente(las entradas digitales tienen prioridad sobre las analógicas)

- Si detectan que alguna variable esta a 1, actualizan la pantalla y si detectan que alguien pulsa un botón, pone a 1 la misma variable que usan las entradas digitales

Las tareas no funcionan por interrupciones. El problema de usar interrupciones era que necesitaba poner ocho entradas esperando a que se activaran y se produjera una interrupción. Por otro lado, dentro de la función que controlaba la acción de salida de dicha interrupción tenía que hacer una división en función de que entrada había sido activada para activar una salida u otra. Esto no me parecía adecuado tener que estar poniendo condiciones para ver que entrada de las posibles era la que había hecho saltar la interrupción.
El código está programado con tareas que son básicamente como pequeños programas que están ejecutándose todo el rato dentro de un bucle infinito. Tienen la característica de que no pueden devolver nada y que si no sirven se deben destruir para que no ocupen memoria. Las tareas requieren RAM que se utiliza para mantener el estado de la tarea, y que la tarea utiliza como su pila. Nosotros podemos crear todas las tareas que queramos, sin embargo, solo podrá estar ejecutándose una en cada momento de tiempo dado que todas se ejecutan en el mismo núcleo de procesador. Se dice que las tareas tienen dos estados distintos: en ejecución o no está en ejecución. En ejecución solo puede haber una tarea mientras que todas las demás se encontrarán en modo no está en ejecución. Mientras una tarea no está en ejecución lo que hace es esperar en el mismo sitio en el que se quedó cuando estaba en ejecución para cuando vuelva a entrar en este estado pueda retomar la tarea por donde la dejó. Para nosotros las tareas parecerá que se están ejecutando en paralelo. Los ciclos se ejecutan a una velocidad que para nuestra percepción veremos que todas las tareas se realizan a la vez.

Ya hemos comentado cuantas tareas tenemos y la función de cada una de las tareas. Se ha hecho de esta manera para intentar que el comportamiento fuera parecido al de un PLC. Normalmente, estos primero toman los datos correspondientes y después actúan en consecuencia a ellos. Para eso, se ha programado las tres tareas encargadas de comprobar si hay algún dato a la entrada y ponen la variable correspondiente a 1 para avisar del evento. Por último, solo hay una tarea encargada de manejar las salidas. En cada iteración comprueba cuales de todas las variables usadas para definir los eventos está a 1. En función de esto, activa la salida correspondiente. Como es esta única función la que decide si se activa una salida y como debe activarse (como salida digital o analógica) simplemente se ha elegido como preferencia que si alguna salida digital está activada las salidas analógicas no pueden. Puede ser que llegue alguna señal analógica pero no se activará en este caso.
Para no tener tantos casos y no hacer que la función que controla las salidas elija entre tres variables(contando con la variable que le correspondería a la pantalla), se ha usado una única variable tanto como para la pantalla como para las entradas digitales ya que realmente su función es la misma.

Para la programación de la pantalla se han usado dos librerías usadas en cursos anteriores. Estas librerías están pensadas para otro microcontrolador así que he tenido que cambiar las funciones correspondientes al control de periféricos por los

periféricos de mi micro. Las otras funciones se basan siempre en las mismas que son las que he tenido que cambiar. Todo lo demás se deja igual y podemos usar la pantalla sin necesidad de crear todas las funciones desde cero.

A continuación podremos ver un diagrama de flujos por cada tarea planteada anteriormente en el que se pueda apreciar los estados por los que tendrá que pasar cada función:

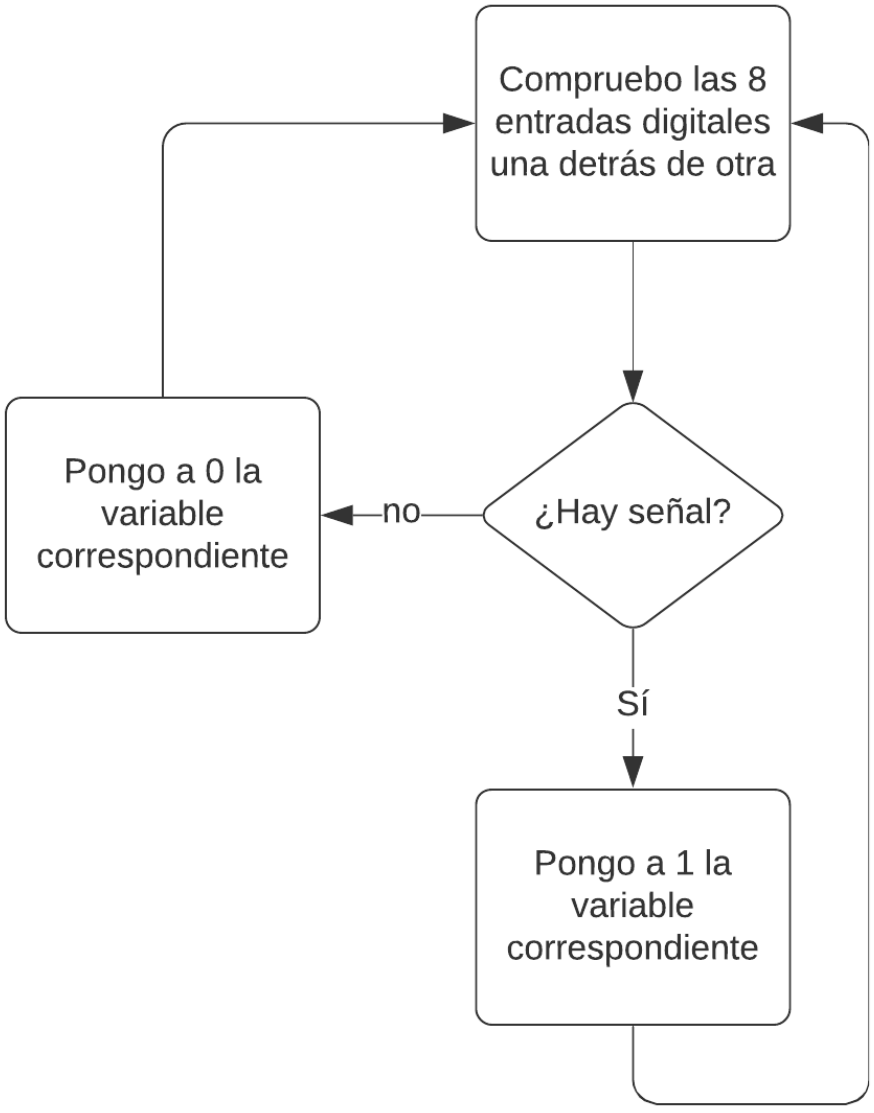


Figura 5.2: Diagrama de flujo detección entradas digitales

Aquí tenemos como es el bucle que realiza la tarea encargada de poner las variables correspondientes a las entradas digitales a 1 cuando hay señal en dichas entradas.

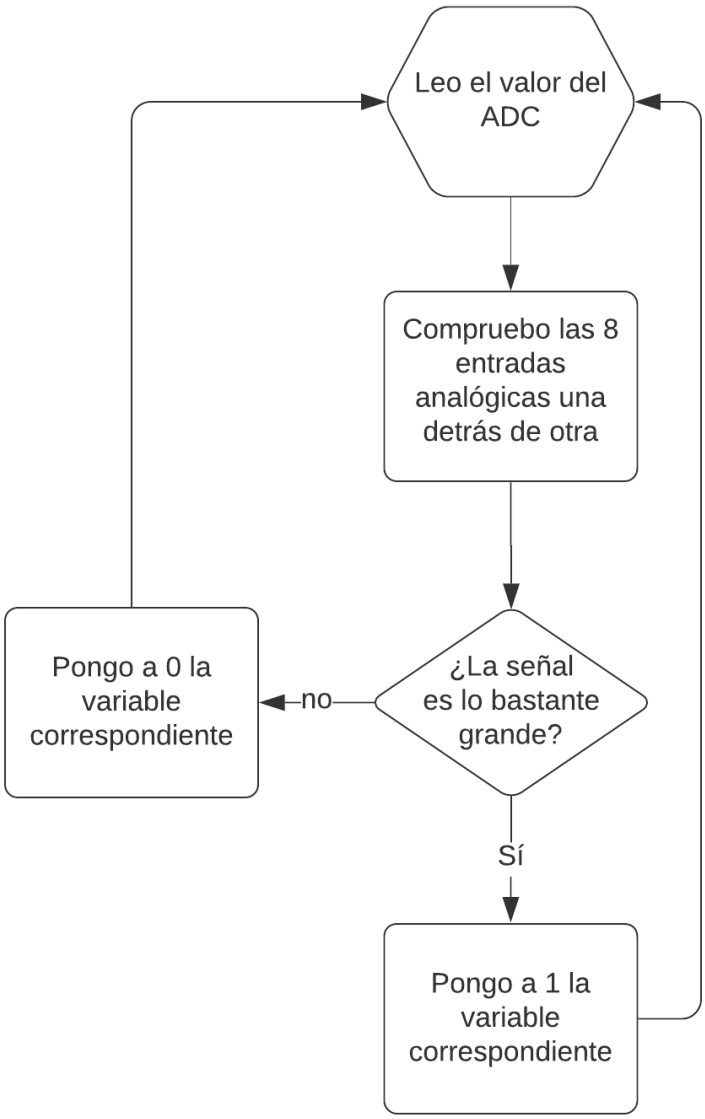


Figura 5.3: Diagrama de flujo detección entradas analógicas

Aquí tenemos como es el bucle que realiza la tarea encargada de poner las variables correspondientes a las entradas analógicas a 1 cuando hay señal en dichas entradas. Los valores de cada entrada se guardan cuando leemos el ADC. La pregunta que hay que hacerse es si la señal es lo bastante grande porque normalmente aunque no le llegue nada siempre lee un valor mayor que 0 por lo que obligamos a que el valor sea mayor que 1000 para poder activarse, teniendo en cuenta que su máximo valor es de 4095.

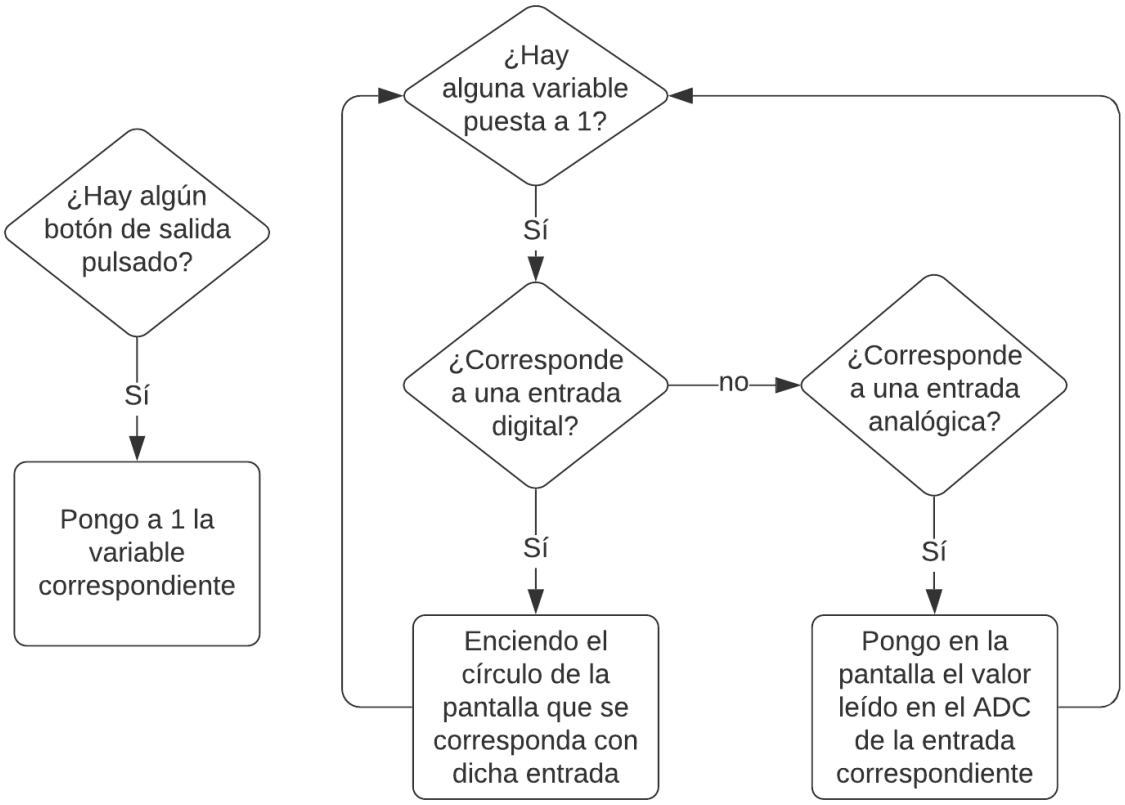


Figura 5.4: Diagrama de flujo controlador de pantalla

Para el control de la pantalla por un lado hay que ver si algún botón está pulsado para activar la variable correspondiente y una vez comprobado esto, tenemos que ver si alguna entrada ha sido activada para pintar la pantalla de nuevo según los datos que hayamos obtenido. Si hay un par de entradas digitales con sus variables a 1 y otro par de entradas analógicas igual, habrá que pintar la nueva pantalla donde aparecerán esas entradas digitales como círculos rojos (mostrado anteriormente en la figura 5.1) y el valor que haya sido leído por el ADC en las variables correspondientes.

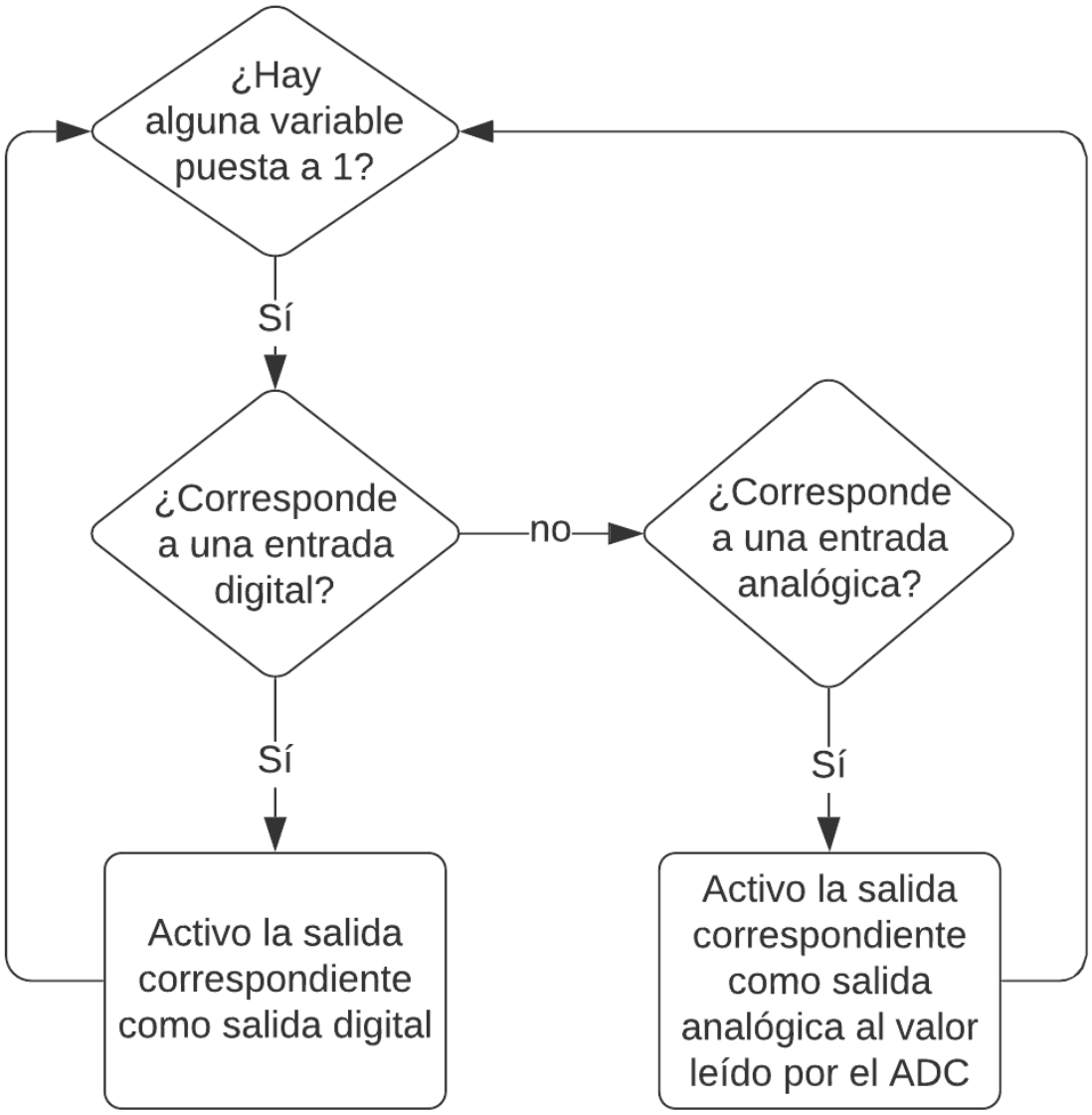


Figura 5.5: Diagrama de flujo controlador de salidas

Por último así es el bucle que activa las salidas en función de las entradas que estén activas, ya sean entradas digitales, analógicas o activadas mediante el botón de la pantalla.

Capítulo 6

Resultados

En este apartado mostraremos el resultado del trabajo completo. Podremos ver como ha quedado finalmente cada placa por separado y como juntas forman todo el sistema equivalente al de un PLC. También discutiremos su funcionamiento y se mostrará una foto donde pueda comprobarse que realmente algún led está encendido como señal de que el sistema funciona correctamente.

Para empezar, a continuación tenemos las fotos de las diferentes placas.

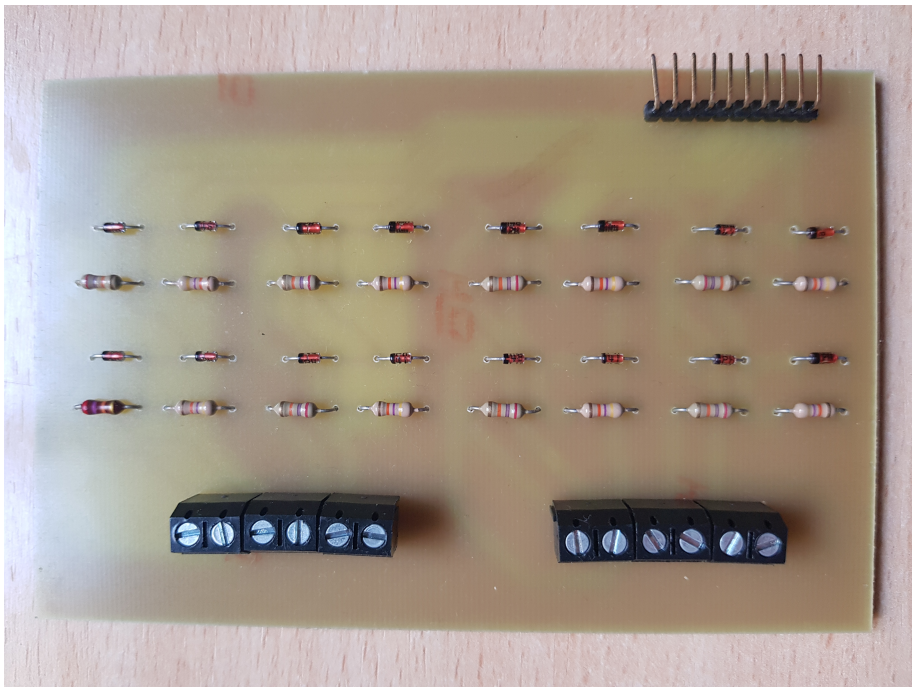


Figura 6.1: Placa original de entradas analógicas

Esta placa como podemos recordar se hizo exclusivamente a una cara. Esta totalmente soldada por debajo. Los conectores que dan al exterior no están totalmente alineados debido a que era mucho más fácil encontrar conectores de dos pines a conectores de ocho por lo que al poner varios de dos juntos están un poco más apretados.

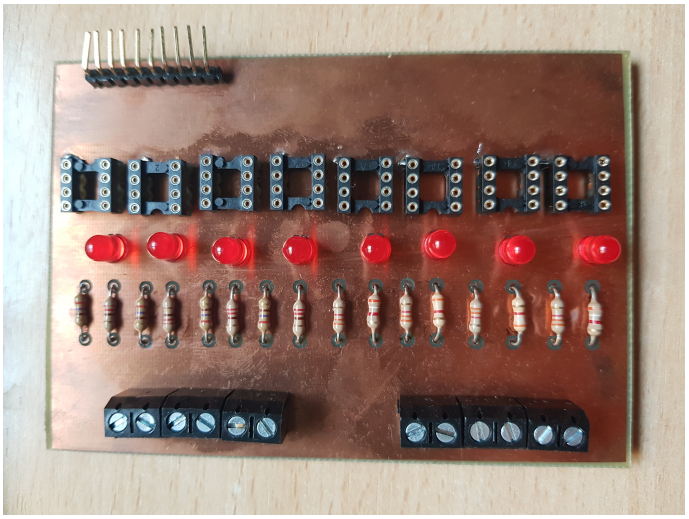


Figura 6.2: Placa original de entradas digitales

Esta placa está hecha a doble capa, de ahí que se vea el cobre por arriba también. Para facilitar la soldadura, lo que está soldado no son los optoacopladores sino unos adaptadores donde poder pinchar los optoacopladores más tarde. Aparte, comentar que realmente por la capa de arriba solo está soldada una pata de cada adaptador. Cada adaptador necesita tener dos patas conectadas a tierra, pero para facilitar la soldadura en la capa superior lo que he hecho es soldar todas las patas del aparato por la capa inferior y conectar solo una a la capa superior. Así, a cada componente le llega la conexión de tierra por la capa superior y esta se conecta a la otra pata que necesita a través de la capa inferior(naturalmente las patas que están soldadas arriba también lo están abajo).

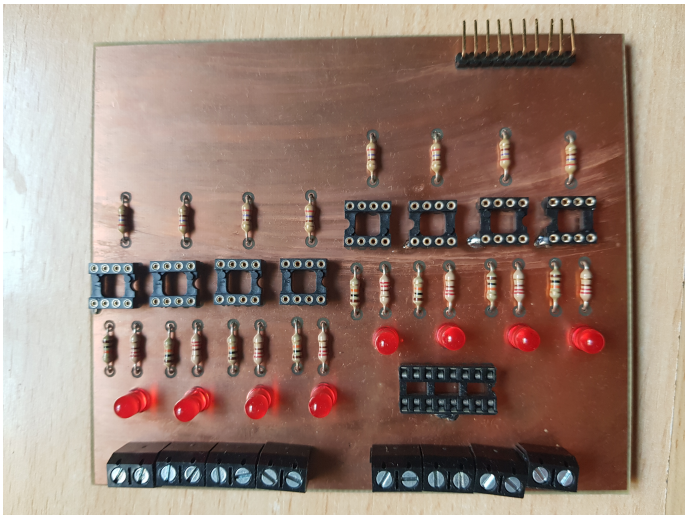


Figura 6.3: Placa original de salidas

Con esta placa pasa exactamente lo mismo que con la placa de entradas digitales. Por culpa del problema de los optoacopladores se tuvo que hacer a doble cara pero se ha seguido el mismo método para facilitar el trabajo de soldadura de la placa.

Además tiene un adaptador más para el L293D el cuál también tiene algunos pines soldados por la capa superior a tierra.

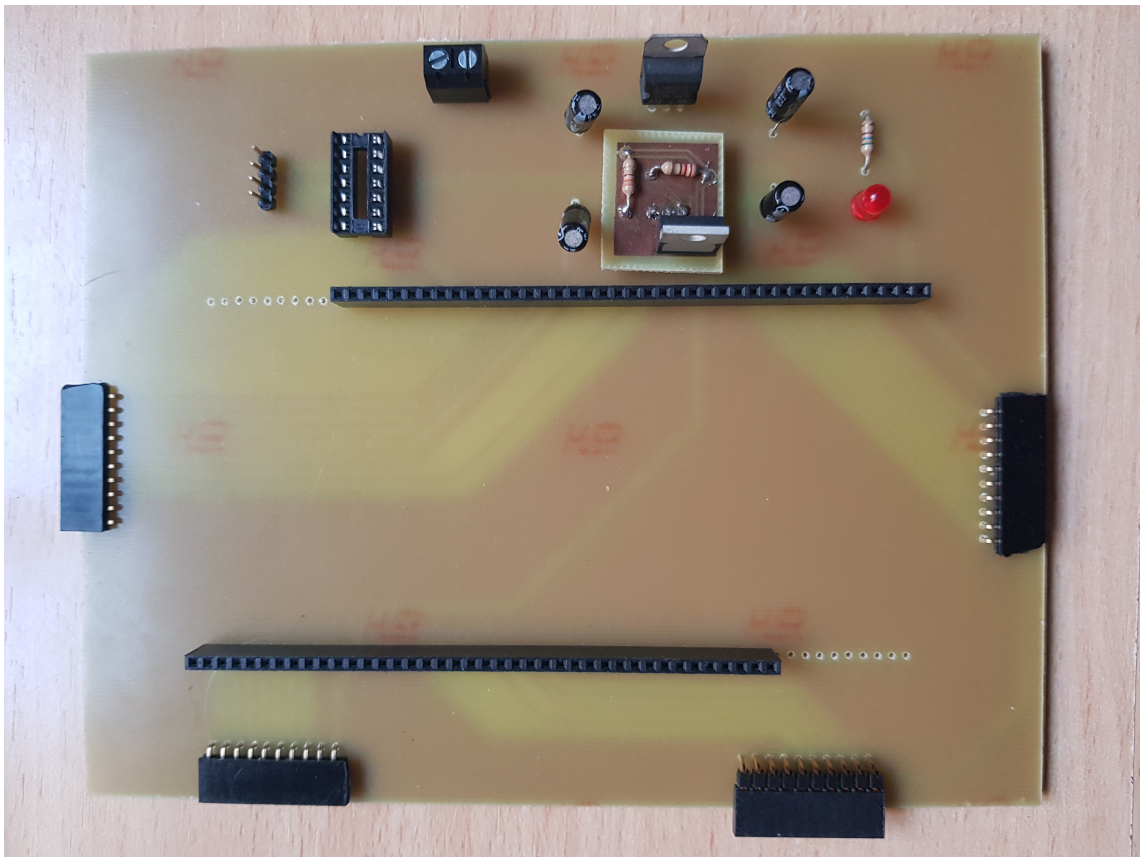


Figura 6.4: Placa original principal

Como podemos ver en esta foto no solo vemos la placa principal sino que también vemos la placa auxiliar soldada a esta misma. La placa auxiliar se encuentra en la parte de arriba de la fotografía donde apreciamos dos resistencias y el LM117 soldados a una pequeña placa que se encuentra entre los condensadores y justo debajo del regulador de voltaje.

Aparte de esto, esta placa también cuenta con un adaptador para el max3079eapd y un conector de 4 pines para el RS-485 que podemos ver cerca del adaptador en la parte superior de la fotografía.

Por último sobre esta placa comentar que para el microcontrolador no se han soldado finalmente conectores de 50 pines, para cubrir ambas partes del microcontrolador sino que solamente habían disponibles conectores de 40 pines. Teniendo en cuenta de que no son necesarios los 50 pines, se han adaptado y a cada lado se ha soldado el conector por la parte que más falta hiciera. De ahí que se vean algunos agujeros en la placa donde no hay conectores.

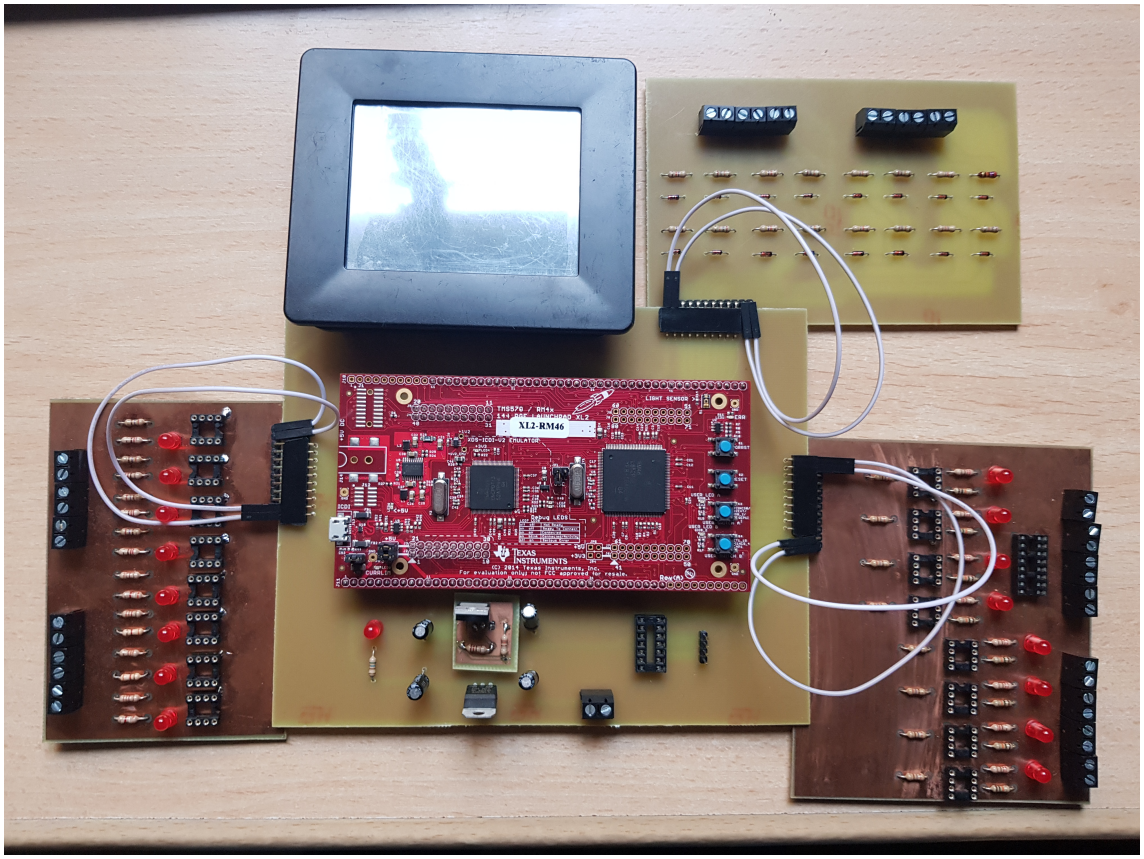


Figura 6.5: Sistema completo (como tendríamos un PLC)

Así quedaría el sistema completo con todas las placas, la pantalla y el microcontrolador unidos. Esto formaría el conjunto de un PLC. A la hora de unir las placas a la placa principal ha habido un problema. Los conectores de la placa principal están del revés. Es decir, para cualquier placa, la distribución era siempre la misma, el pin de tierra y el de alimentación(no tenía porque ser siempre en ese orden) y después los pines restantes. La placa principal tiene los pines de alimentación y tierra justo en el vértice opuesto que todas las demás placas. Para solucionar este problema, lo que se ha hecho es conectar directamente todos los pines menos estos dos mencionados y a través de dos cables que se pueden ver perfectamente en la imagen conectar los 2 pines restantes de cada placa. Otra opción habría sido poner del revés o la placa principal o todas las demás placas pero se descartó esta idea por el hecho de que todas las placas excepto la de entradas analógicas tienen leds. Si le daba la vuelta a cualquiera de las placas se pierde la visión de los leds por lo que no tendría sentido que estuvieran.

Por la parte de abajo de la figura está el conector para la alimentación del sistema. Tanto la alimentación como las entradas y salidas no se contemplan en esta imagen dado que no son parte de un PLC si no que se obtienen del exterior. Más adelante cuando se muestre la figura con el PLC en funcionamiento se podrá ver con todos los componentes a la vez y con la pantalla funcionando.

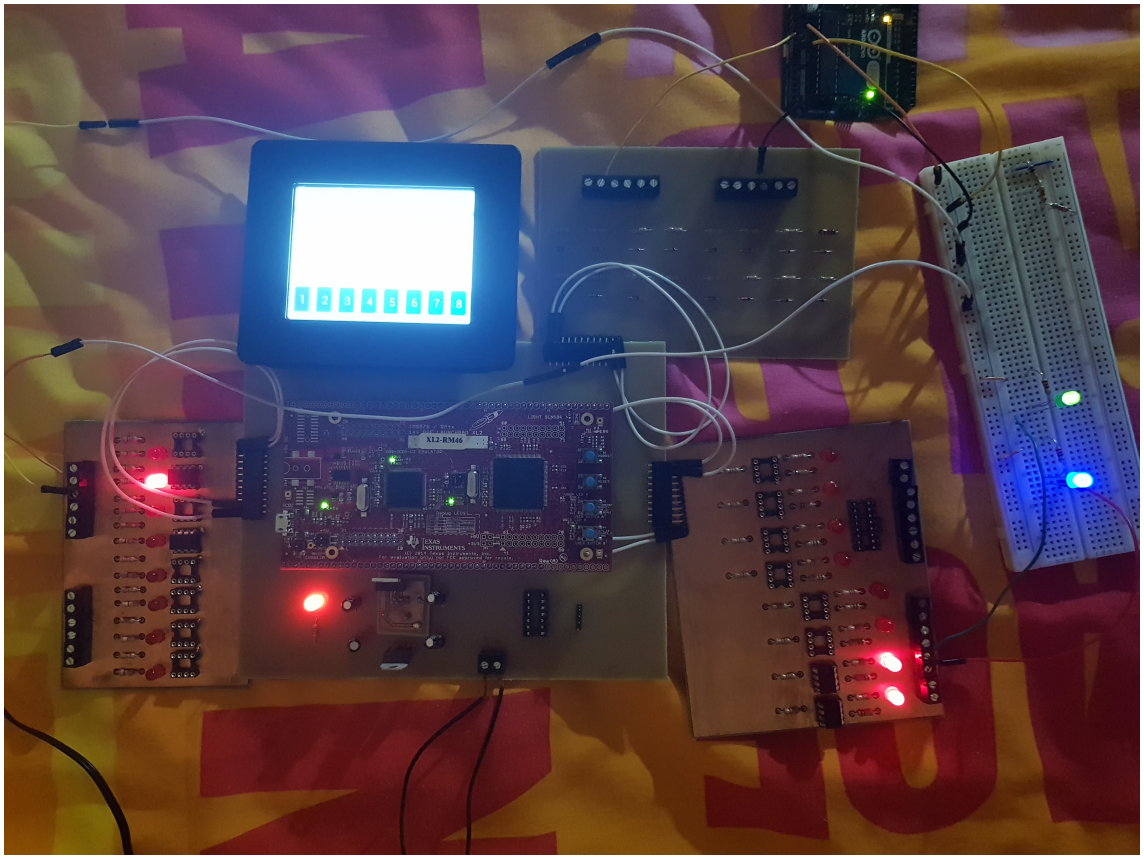


Figura 6.6: Sistema completo funcionando

Aquí podemos ver finalmente como es el sistema completo funcionando. En esta primera fotografía se aprecian mejor las conexiones. Por falta de material (optoacopladores y fuente de alimentación de 24V), solo hemos probado dos entradas y dos salidas. En la imagen podemos ver como las conexiones de las entradas digitales se corresponden justo al contrario con las salidas, es decir, la entrada que se puede ver más arriba se corresponde con la salida que en la foto ocupa la posición más baja.
Para las alimentaciones de las entradas se ha usado una placa arduino básica, la cuál ni siquiera hemos programado, simplemente hemos usado las alimentaciones de 3,3V y de 5V que tiene a su disposición. Con la de 3,3V hemos simulado una entrada digital, y con la de 5V una entrada analógica consiguiendo ver así las dos salidas correspondientes. Para poder usar este voltaje que nos proporciona la placa arduino, tanto en la placa de entradas analógicas como en la de entradas digitales hemos conectado el cable de tierra del arduino a nuestro sistema. Realmente bastaría con que fuera solo en uno pero parece más lógico que cada uno tenga su propia conexión para hacer más evidente la separación.

Un detalle importante y que no se aprecia en la foto es que el led azul que se observa en la protoboard, correspondiente a la última salida, realmente está parpadeando ya que se ha encendido debido a que hay una entrada analógica.
En esta foto no se aprecia muy bien la pantalla por ello hay otra foto para que se pueda comprobar que cada salida está siendo activada por una placa diferente.

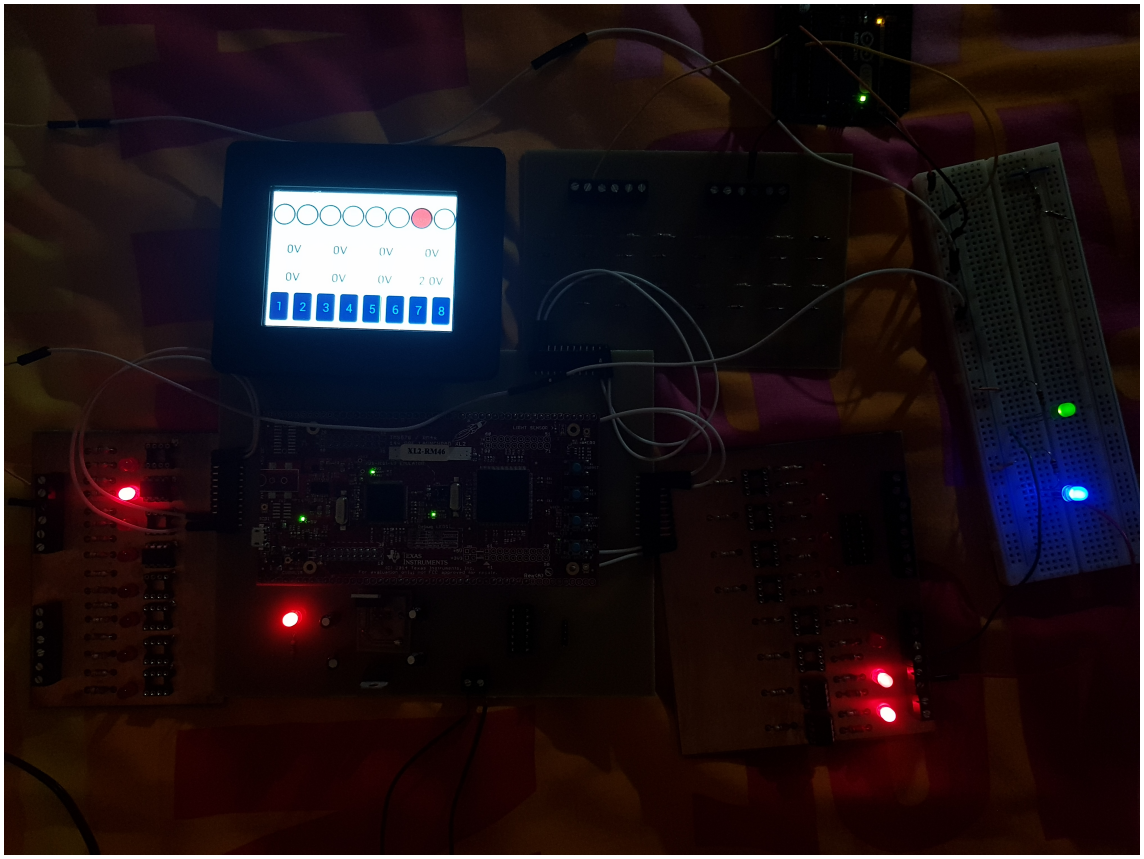


Figura 6.7: Sistema completo funcionando 2

En esta foto se aprecia claramente la pantalla y todos los led que deben estar encendidos.
Aunque está un poco pequeño, en la pantalla podemos ver a parte del círculo rojo correspondiente a la entrada digital, que en el último valor numérico pone 2.0V, correspondiente a a entrada analógica.

El diseño final de la pantalla es un poco distinto al que se presentó en el apartado de programación del microcontrolador. Simplemente en vez de poner todos los valores numéricos de las entradas analógicas, me ha parecido más conveniente separarlo en dos tandas de cuatro. Los cuatro de arriba van de la entrada número 1 a la 4 (de izquierda a derecha) y en la parte de abajo tenemos de la 5 a la 8. En la figura 6.6 de la página anterior se ve mejor pero aún así también en esta podemos ver como es el orden que sigue la placa de entradas analógicas, siendo la que está más a la derecha en la foto la última.

Como se ha comentado anteriormente, por falta de material y de fuentes de alimentación no se ha podido probar las entradas y salidas que se conectarían a 24V en ves de a 3,3V. De todas formas, el comportamiento es análogo y funcionaría de la misma manera. Por otro lado, comentar algunos de los inconvenientes que se han sufrido.

En primer lugar, al final en lm1117 no necesitaba placa de adaptación. Es cierto que aún así la huella de KiCad no era correcta y había que hacer un cambio entre los pines del componente pero el lm117 que se ha usado era específico de salida 3,3V por lo que no hacia falta ponerle las resistencias para regular la tensión a la salida. Para no tener que desmontar la placa de adaptación que ya estaba soldada lo que se ha hecho es quitar las resistencias que había y soldar la pata del componente que antes iba conectada al pequeño circuito directamente a tierra. Me di cuenta de este fallo cuando probando las distintas conexiones ví que a la placa de las salidas le estaba llegando 10V en vez de 3,3V. Una vez que se localizó el problema la solución fue bastante fácil. En la figura 6.6 se puede ver como faltan esas dos resistencias que si se aprecian en la figura 6.5. Fue un inconveniente que se encontró justo al final cuando se probó el sistema al completo y se ha resuelto de la manera más simple y rápida.

El segundo inconveniente es que después de un buen rato usando el sistema parece que el componente 7805 no funciona bien y tanto la pantalla como el propio microcontrolador (los dos componentes que se alimentan directamente desde el componente) dejan de funcionar entrando el Hercules en un estado de error. Este problema solo ocurre cuando la pantalla esta conectada. Si desconecto la pantalla el plc puede seguir funcionando todo el tiempo.

Algo a tener en cuenta es que el código que sigue el microcontrolador antes de crear todas la tareas calibra la pantalla. Esto se hace como parte de una iniciación donde se inician todos los periféricos también. Por lo tanto, lo primero que tenemos que hacer es seguir los pasos que nos muestra la pantalla para ajustar el calibrado y una vez hecho aparece nuestro dibujo principal en la pantalla y a partir de ese momento queda iniciado el plc. Se debe tener en cuenta porque si queremos usar el sistema primero debemos conectar la pantalla para hacer esta iniciación y luego para que no de problemas podemos desconectarla y seguir usando todo el sistema sin la pantalla.
Esta calibración de la pantalla no se menciona en el apartado de programación primero porque no es un código que haya hecho yo y segundo porque en principio pensé que no haría falta. Luego cuando probé la pantalla ví que sin una calibración previa, cuando intentabas pulsar los botones, se creía que estabas pulsando otros distintos, como si estuvieran desplazados y parte de la pantalla ni siquiera se pudiera leer. En el código que viene a continuación se puede ver que está en la función principal antes incluso de crear las tareas.

Estos son los resultados finales de este trabajo.

Capítulo 7

Anexo

```
#include "system.h"
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>
#include <stdbool.h>
#include "sys_common.h"
#include "stdlib.h"
#include "FreeRTOS.h"
#include "os_task.h"
#include "os_semphr.h"

#include "sci.h"
#include "adc.h"
#include "het.h"
#include "gio.h"
#include "etpwm.h"
#include "FT800.h"
#include "mibspi.h"

#include "sys_common.h"


// Declaración de variables
adcData_t adc_data[8];

char chipid = 0;                          // Holds value of Chip
    ID read from the FT800

unsigned long cmdBufferRd = 0x00000000;          // Store the
    value read from the REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;          // Store the
    value read from the REG_CMD_WRITE register
unsigned int t=0;

int Fin_Rx=0;
char Buffer_Rx;
unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
```

```
char cadena1[5];

float voltage1;

bool input_1;
bool input_2;
bool input_3;
bool input_4;
bool input_5;
bool input_6;
bool input_7;
bool input_8;

bool input_dig1;
bool input_dig2;
bool input_dig3;
bool input_dig4;
bool input_dig5;
bool input_dig6;
bool input_dig7;
bool input_dig8;

bool input_an1;
bool input_an2;
bool input_an3;
bool input_an4;
bool input_an5;
bool input_an6;
bool input_an7;
bool input_an8;


xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;
xTaskHandle xTask3Handle;
xTaskHandle xTask4Handle;
xTaskHandle xTask5Handle;

///* Definición de las tareas */
void vTask1(void *pvParameters)
{
while(1)
{
// Si encuentra alguna entrada a 0(pullup), activa la variable
    correspondiente
if(gioGetBit(gioPORTB, 2)==0)input_dig1 = true;
else input_dig1 = false;

if(gioGetBit(gioPORTB, 3)==0) input_dig2 = true;
else input_dig2 = false;

if(gioGetBit(gioPORTA, 0)==0) input_dig3 = true;
else input_dig3 = false;

if(gioGetBit(hetPORT1, 29)==0) input_dig4 = true;
```

```
else input_dig4 = false;

if(gioGetBit(hetPORT1, 27)==0) input_dig5 = true;
else input_dig5 = false;

if(gioGetBit(gioPORTA, 1)==0) input_dig6 = true;
else input_dig6 = false;

if(gioGetBit(hetPORT1, 11)==0) input_dig7 = true;
else input_dig7 = false;

if(gioGetBit(gioPORTA, 2)==0) input_dig8 = true;
else input_dig8 = false;
}
}

void vTask2(void *pvParameters)
{

while(1)
{
MPU_vTaskDelay(100);
// Leemos el convertidor analógico digital
adcStartConversion(adcREG1, adcGROUP1);
while(!adcIsConversionComplete(adcREG1, adcGROUP1));
adcGetData(adcREG1, adcGROUP1, &adc_data[0]);

// Si encuentra algún valor notablemente alto activa la variable
    correspondiente
if((unsigned int)(adc_data[3].value)>1500) input_an1 = true;
else input_an1 = false;

if((unsigned int)(adc_data[4].value)>1500) input_an2 = true;
else input_an2 = false;

if((unsigned int)(adc_data[1].value)>1500) input_an3 = true;
else input_an3 = false;

if((unsigned int)(adc_data[0].value)>1500) input_an4 = true;
else input_an4 = false;

if((unsigned int)(adc_data[5].value)>1500) input_an5 = true;
else input_an5 = false;

if((unsigned int)(adc_data[7].value)>1500) input_an6 = true;
else input_an6 = false;

if((unsigned int)(adc_data[6].value)>1500) input_an7 = true;
else input_an7 = false;

if((unsigned int)(adc_data[2].value)>1500) input_an8 = true;
else input_an8 = false;
}
}

void vTask3(void *pvParameters)
```

```
{
while(1)
{
// Activa las salidas en función de las entradas leídas
if(input_dig1 == true || input_1 == true) etpwmSetCmpB(etpwmREG1
    , 4095);
else if(input_an1 == true) etpwmSetCmpB(etpwmREG1, (unsigned int
    )(adc_data[3].value));
else etpwmSetCmpB(etpwmREG1, 0);

if(input_dig2 == true || input_2 == true) etpwmSetCmpB(etpwmREG2
    , 4095);
else if(input_an2 == true) etpwmSetCmpB(etpwmREG2, (unsigned int
    )(adc_data[4].value));
else etpwmSetCmpB(etpwmREG2, 0);

if(input_dig3 == true || input_3 == true) etpwmSetCmpA(etpwmREG3
    , 4095);
else if(input_an3 == true) etpwmSetCmpA(etpwmREG3, (unsigned int
    )(adc_data[1].value));
else etpwmSetCmpA(etpwmREG3, 0);

if(input_dig4 == true || input_4 == true) etpwmSetCmpB(etpwmREG3
    , 4095);
else if(input_an4 == true) etpwmSetCmpB(etpwmREG3, (unsigned int
    )(adc_data[0].value));
else etpwmSetCmpB(etpwmREG3, 0);

if(input_dig5 == true || input_5 == true) etpwmSetCmpA(etpwmREG4
    , 4095);
else if(input_an5 == true) etpwmSetCmpA(etpwmREG4, (unsigned int
    )(adc_data[5].value));
else etpwmSetCmpA(etpwmREG4, 0);

if(input_dig6 == true || input_6 == true) etpwmSetCmpB(etpwmREG4
    , 4095);
else if(input_an6 == true) etpwmSetCmpB(etpwmREG4, (unsigned int
    )(adc_data[7].value));
else etpwmSetCmpB(etpwmREG4, 0);

if(input_dig7 == true || input_7 == true) etpwmSetCmpB(etpwmREG7
    , 4095);
else if(input_an7 == true) etpwmSetCmpB(etpwmREG7, (unsigned int
    )(adc_data[6].value));
else etpwmSetCmpB(etpwmREG7, 0);

if(input_dig8 == true || input_8 == true) etpwmSetCmpA(etpwmREG7
    , 4095);
else if(input_an8 == true) etpwmSetCmpA(etpwmREG7, (unsigned int
    )(adc_data[2].value));
else etpwmSetCmpA(etpwmREG7, 0);

MPU_vTaskDelay(50);
}
}
```

```
void vTask4(void *pvParameters)
{
while(1)
{
// Leemos la pantalla y si hay algun botón pulsado se activa la
   variable
Lee_pantalla();

if(POSX>2 && POSX<38 && POSY>180 && POSY<220) input_1 = true;
else input_1 = false;
if(POSX>42 && POSX<78 && POSY>180 && POSY<220) input_2 = true;
else input_2 = false;
if(POSX>82 && POSX<118 && POSY>180 && POSY<220) input_3 = true;
else input_3 = false;
if(POSX>122 && POSX<158 && POSY>180 && POSY<220) input_4 = true;
else input_4 = false;
if(POSX>162 && POSX<198 && POSY>180 && POSY<220) input_5 = true;
else input_5 = false;
if(POSX>202 && POSX<238 && POSY>180 && POSY<220) input_6 = true;
else input_6 = false;
if(POSX>242 && POSX<278 && POSY>180 && POSY<220) input_7 = true;
else input_7 = false;
if(POSX>282 && POSX<318 && POSY>180 && POSY<220) input_8 = true;
else input_8 = false;

// Pintamos la nueva pantalla
Nueva_pantalla(0xff,0xff,0xff);

// Botones de salida
ComColor(0,0,0xff);
ComButton(4,180,32,50,28,256,"1");
ComButton(44,180,32,50,28,256,"2");
ComButton(84,180,32,50,28,256,"3");
ComButton(124,180,32,50,28,256,"4");
ComButton(164,180,32,50,28,256,"5");
ComButton(204,180,32,50,28,256,"6");
ComButton(244,180,32,50,28,256,"7");
ComButton(284,180,32,50,28,256,"8");

// Bordes negros de los círculos
ComColor(0,0,0);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(20,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(60,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(100,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(140,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(180,40);
```

```
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(220,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(260,40);
ComPointSize(19);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(300,40);

// Segun si esta la salida a 1 o no rellenamos de blanco o de
   rojo
if(input_dig1 == true || input_1 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(20,40);

if(input_dig2 == true || input_2 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(60,40);

if(input_dig3 == true || input_3 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(100,40);

if(input_dig4 == true || input_4 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(140,40);

if(input_dig5 == true || input_5 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(180,40);

if(input_dig6 == true || input_6 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(220,40);

if(input_dig7 == true || input_7 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(260,40);

if(input_dig8 == true || input_8 == true) ComColor(0xff,0,0);
else ComColor(0xff,0xff,0xff);
```

```
ComPointSize(17);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(300,40);

// Texto con el valor de analógica
ComColor(0,0,0);
if(input_an1 == true)
{
voltage1 = (adc_data[3].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(40,90,28,OPT_CENTERX,cadena1);
}
else ComTXT(40,90,28,OPT_CENTERX,"0V");

if(input_an2 == true)
{
voltage1 = (adc_data[4].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(120,90,28,OPT_CENTERX,cadena1);
}
else ComTXT(120,90,28,OPT_CENTERX,"0V");

if(input_an3 == true)
{
voltage1 = (adc_data[1].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(200,90,28,OPT_CENTERX,cadena1);
}
else ComTXT(200,90,28,OPT_CENTERX,"0V");

if(input_an4 == true)
{
voltage1 = (adc_data[0].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(280,90,28,OPT_CENTERX,cadena1);
}
else ComTXT(280,90,28,OPT_CENTERX,"0V");

if(input_an5 == true)
{
voltage1 = (adc_data[5].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(40,140,28,OPT_CENTERX,cadena1);
}
else ComTXT(40,140,28,OPT_CENTERX,"0V");

if(input_an6 == true)
{
voltage1 = (adc_data[7].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(120,140,28,OPT_CENTERX,cadena1);
}
else ComTXT(120,140,28,OPT_CENTERX,"0V");

if(input_an7 == true)
{
```

```
voltage1 = (adc_data[6].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(200,140,28,OPT_CENTERX,cadena1);
}
else ComTXT(200,140,28,OPT_CENTERX,"0V");

if(input_an8 == true)
{
voltage1 = (adc_data[2].value)/1240;
sprintf(cadena1, "%.1fV", voltage1);
ComTXT(280,140,28,OPT_CENTERX,cadena1);
}
else ComTXT(280,140,28,OPT_CENTERX,"0V");

// Dibuja el resultado final
Dibuja();

}
}


// Función principal
int main(void)
{
int i;

gioInit();
hetInit();
adcInit();
sciInit();
etpwmInit();
mibspiInit();

// Configuramos los pwm de las salidas
etpwmSetClkDiv(etpwmREG1, ClkDiv_by_128, HspClkDiv_by_14);
etpwmSetTimebasePeriod(etpwmREG1, 4095);
etpwmSetClkDiv(etpwmREG2, ClkDiv_by_128, HspClkDiv_by_14);
etpwmSetTimebasePeriod(etpwmREG2, 4095);
etpwmSetClkDiv(etpwmREG3, ClkDiv_by_128, HspClkDiv_by_14);
etpwmSetTimebasePeriod(etpwmREG3, 4095);
etpwmSetClkDiv(etpwmREG4, ClkDiv_by_128, HspClkDiv_by_14);
etpwmSetTimebasePeriod(etpwmREG4, 4095);
etpwmSetClkDiv(etpwmREG7, ClkDiv_by_128, HspClkDiv_by_14);
etpwmSetTimebasePeriod(etpwmREG7, 4095);
etpwmStartTBCLK();

// Primero calibramos la pantalla
Inicia_pantalla();

Nueva_pantalla(0x10,0x10,0x10);


ComColor(21,160,6);
ComLineWidth(5);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(10,10);
```

```
ComVertex2ff(310,230);
ComColor(65,202,42);
ComVertex2ff(12,12);
ComVertex2ff(308,228);


Comando(CMD_END);
ComColor(0xff,0xff,0xff);
ComTXT(160,50, 22, OPT_CENTERX,"CALIBRACION de PANTALLA");
ComTXT(160,100, 22, OPT_CENTERX," TFG 4GIERM 2020 ");
ComTXT(160,150, 22, OPT_CENTERX," Toca la pantalla para seguir "
    );
Comando(CMD_BEGIN_LINES);
ComVertex2ff(40,40);
ComVertex2ff(280,40);
ComVertex2ff(280,40);
ComVertex2ff(280,200);
ComVertex2ff(280,200);
ComVertex2ff(40,200);
ComVertex2ff(40,200);
ComVertex2ff(40,40);
Comando(CMD_END);

Dibuja();
Espera_pant();

Nueva_pantalla(0xf0,0xf0,0x00);


ComTXT(60,30,27,0,"Pulsa en el punto");
EscribeRam32(CMD_CALIBRATE);

Dibuja();
ComEsperaFin();

for(i=0;i<6;i++)     REG_TT[i]=Lee_Reg(REG_TOUCH_TRANSFORM_A+4*i)
    ;

// Creamos las tareas
if (MPU_xTaskCreate(vTask1,"InputDig", configMINIMAL_STACK_SIZE,
     NULL, 1, &xTask1Handle) != pdTRUE)
{
/* La tarea no pudo ser creada */
while(1);
}

if (MPU_xTaskCreate(vTask2,"InputAn", configMINIMAL_STACK_SIZE,
    NULL, 1, &xTask2Handle) != pdTRUE)
{
/* La tarea no pudo ser creada */
while(1);
}

if (MPU_xTaskCreate(vTask3,"Output", configMINIMAL_STACK_SIZE,
    NULL, 1, &xTask3Handle) != pdTRUE)
{
```

```
/* La tarea no pudo ser creada */
while(1);
}

if (MPU_xTaskCreate(vTask4,"ft800", configMINIMAL_STACK_SIZE,
    NULL, 1, &xTask4Handle) != pdTRUE)
{
/* La tarea no pudo ser creada */
while(1);
}

/* Damos inicio a las tareas */
vTaskStartScheduler();


/* Run forever */
while(1);

}
```

Bibliografía

[1] Texas Instruments Junio 2015, *RM46L852 16- and 32-Bit RISC Flash Microcontroller*. Recuperado de http://www.ti.com/lit/ds/spns185c/spns185c.pdf.

[2] KiCad 22 Enero 2019, *Comenzando en Kicad*. Recuperado de https://docs.kicad-pcb.org/4.0.7/es/getting-started-in-kicad/getting-started-in-kicad.pdf.

[3] PLC:Controlador Lógico Programable, *Historia*. Recuperado de http://controladoreslocgicosprogramables.blogspot.com/p/historia.html.

[4] FTDI Chip 1-Julio-2014, *FT800 Series Programmer Guide*. Recuperado de FT800 Programmers Guide.pdf.

[5] Vishay Semiconductors 27-Septiembre-16, *Datasheet del componente 6n137*. Recuperado de https://www.vishay.com/docs/84732/6n137.pdf.

[6] Texas Instruments Enero 2016, *Datasheet del componente LM1117*. Recuperado de http://www.ti.com/lit/ds/symlink/lm1117.pdf.

[7] Fairchild Semiconductor, *Datasheet del componente LM7805*. Recuperado de https://html.alldatasheet.es/html-pdf/82833/FAIRCHILD/LM7805/405/1/LM7805.html.

[8] Maxim Integrated Enero 2016, *MAX3070E–MAX3079E*. Recuperado de https://datasheets.maximintegrated.com/en/ds/MAX3070E-MAX3079E.pdf.

[9] Piensa 3D, *Qué es un optoacoplador, funcionamiento y aplicaciones*. Recuperado de https://piensa3d.com/que-es-un-optoacoplador-funcionamiento-aplicaciones/.

[10] Resistor Guide, *Resistor Values*. Recuperado de http://www.resistorguide.com/resistor-values/E12-series-tolerance-10.

[11] Designer Edge, *Hercules Launchpad*. Recuperado de https://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/6494/Hercules-Launchpad.aspx.

[12] FreeRTOS, *TI Hercules Safety MCUs RTOS Demo*. Recuperado de https://www.freertos.org/Free-RTOS-for-TI-RM48-and-TMS570.html.